AN IMMERSIVE SYNTHETIC ENVIRONMENT FOR OBSERVATION AND INTERACTION WITH A LARGE VOLUME OF INTEREST

THESIS

Rex G Haddix II, Capt, USAF

AFIT/GCS/ENG/93M-02

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/93M-02

AN IMMERSIVE SYNTHETIC ENVIRONMENT FOR OBSERVATION AND INTERACTION WITH A LARGE VOLUME OF INTEREST

THESIS

Rex G. Haddix II, Capt, USAF

AFIT/GCS/ENG/93M-02

AFIT/GCS/ENG/93M-02

AN IMMERSIVE SYNTHETIC ENVIRONMENT FOR OBSERVATION AND INTERACTION WITH A LARGE VOLUME OF INTEREST

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Computer Systems

Rex G. Haddix II

Captain, USAF

March 1993

Approved for public release; distribution unlimited

Preface

I am indebted to several people for their assistance and cooperation with this project. In particular, I would like to thank Capt Dave Tisdale, for his insight on working with several of the peripheral devices used in synthetic environments. I would also like to thank Capt David Pond and Capt Bruce Hobbs, for their assistance in working with the synthetic environment's design and operation. I also want to thank LtCol Martin Stytz for his guidance and understanding; he continously provided new avenues to explore. I am also indebted to LtCol Phil Amburn, for his insightful discussions in handling several problems. Special thanks are also extended to Maj Dave Neyland at DARPA and Capt Tip Cliff at JNIDS for their support.

I especially wish to thank my wife, Kathy, and my children, Thea and Jocelyn, for their patience, understanding, and love.

Rex G. Haddix II

Table of Contents

List of Figures

Figure

Abstract

Unlike military campaigns throughout history, modern military commanders are faced with an overwhelming amount of intelligence data concerning the disposition of friendly and non-friendly forces. The sheer volume of data produced by today's automated and semi-automated collection systems make it virtually impossible to process and analyze the data in real-time. What the commander needs is a real-time, three-dimensional representation of the battlefield with the current disposition of all forces. This thesis addresses the initial development of a Synthetic Battlebridge System designed to provide the user with a synthetic three-dimensional view of moving and stationary vehicles dispersed over a hundred thousand cubic mile volume. The system contains provisions to allow a user to view the battlefield either in the Polhemus LookingGlass™ fiber optic based high resolution head mounted display, a standard CRT, or through the Fake Space Labs BOOM2M™ high resolution monochrome display. Users can also video tape a session in the Synthetic Battlebridge System. A voice recognition system provides user interaction to the Synthetic Battlebridge System. Using a positional tracker to monitor position and direction of view, the system allows the user to move around and through the battlefield. The system also allows the user to identify and select up to ten different views of the battlefield.

This is the first year of a continuing effort at the Air Force Institute of Technology (AFIT) towards creating a Synthetic Battlebridge System commanders can use to better understand the spatial orientation and spatial distribution of elements on the battlefield. The system will provide increased situational awareness to assist the commander's critical decision making processes.

AN IMMERSIVE SYNTHETIC ENVIRONMENT FOR OBSERVATION AND INTERACTION WITH A LARGE VOLUME OF INTEREST

I. Introduction

1.1 Background

Contrary to most expectations, the end of the Cold War has not lessened international tensions. In fact, the disintegration of many necessary and forced alliances has renewed long standing religious and ethnic hatred and increased clan rivalries and nationalistic instincts throughout the world. Each of these activities can result in potential hot spots of civic unrest that can threaten US and allied interests.

As military and civilian planners develop scenarios to contend with these threats, they are forced to consider the use of armed forces covering the spectrum from full scale war such as used in Desert Shield and Desert Storm to humanitarian relief efforts as those used in Somalia. This unprecedented use of armed forces coupled with decreasing defense expenditures has forced military leaders to seek innovative and economical alternatives to tactical battlefield analysis and mission planning and training systems. One area receiving increased attention is the use of virtual reality or synthetic environments.

Synthetic environments have gained notoriety in the commercial sectors where they have been used for interactive walkthroughs of architectural models (Airey and others, 1990; Funkhouser and others, 1992; Teller and Sequin, 1991), analysis of molecular docking (Ming and others, 1988), planetary surface exploration (Hitchner, 1992), and aircraft airflow dynamics (Levit and Bryson, 1992). Military applications have primarily concentrated on distributed simulations (Blau and others, 1992; Zyda and others 1992), low-cost flight simulators (McCarty, 1993), and satellite modeling and orbital analysis (Pond, 1992).

Synthetic environments can not solve every problem. They can work well in applications involving three-dimensional structures and graphics not requiring high levels of detail. Architectural walkthroughs, molecular modeling, three-dimensional games, and simple sculpting are examples. Although each of the above applications is in the seminal stages of this technology, each demonstrates that synthetic environments can provide an economical and practical alternative, albeit in some cases the only alternative, to the scientific analysis of large volumes of data.

1.2 Problem Statement

Unlike military campaigns throughout history, modern military commanders are faced with an almost overwhelming amount of intelligence data concerning the disposition of friendly and non-friendly forces. The sheer volume of data produced by today's automated and semiautomated collection systems makes it virtually impossible to process and analyze the data in real-time. This places the intelligence community in the tenuous position of filtering the data before releasing it to field commanders. Even after this filtering process is completed, commanders are typically inundated with information in a variety of formats including satellite and airborne reconnaissance imagery, analysts and field reports, and communication and electromagnetic emanation activity logs. Not only is this data untimely, it does not adequately address the key issue for any battlefield commander: Where are my forces? Where are my opponent's forces? What are they doing. The military commander needs a real-time three-dimensional representation of the battlefield with the current disposition of all forces.

1.3 Scope

The primary purpose of this research was to develop an immersive synthetic environment that would provide a user with a three-dimensional representation of moving and stationary vehicles dispersed over a large area of terrain. The environment

accommodates inc[illegible]g levels of resolution for both the terrain and the vehicles of interest and provid[illegible]user with an intuitive and modifiable interface.

This system, the Synthetic Battlebridge System, supports current SIMNET protocol standards and is able to implement Distributive Interactive Simulation (DIS) protocol standards with minimal modifications. This system can also function with a variety of commercially available display and tracking systems. A rapid prototype of the system was built using object-oriented design and programming methods.

This is the first year of a continuing effort at developing such an environment. This effort is jointly sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Joint National Intelligence Defense Staff (JNIDS).

1.4 Complimentary Efforts

Three separate, complementary thesis efforts at AFIT are used in support of this thesis effort:

1. Captain Bruce Hobb's thesis effort resulted in a system to interface and control a real-time true three-dimensional viewing system, the Texas Instruments' OmniView (Hobbs, 1992). Many of the initial design aspects of both Captain Hobbs' work and the Synthetic Battlebridge System were collectively developed.

2. Captain Dave Pond developed a synthetic environment for satellite modeling and analysis (Pond, 1992) Most of the graphics rendering software and some of the voice recognition software was jointly written.

3. Captain Steven Sheasby designed and implemented the network interface for the SIMNET protocol (Sheasby, 1992). All the access methods to allow the Synthetic Battlebridge System to identify and locate vehicles were mutually developed.

Additional support from previous AFIT work included Captain Tisdale's MiniVas controller (Tisdale, 1992), Captain Simpson's RS-232 port drivers (Simpson, 1991), Captain Gerken's Polhemus class structure (Gerken, 1991), Captain Brunderman's

Geometry class structure (Brunderman, 1991), and Captains Wright's and Soltz's sound generation facility for the Macintosh computer (Wright and Soltz, 1992).

1.5 Approach and Methodology

The first step in conducting this research was a literature review of recent developments in synthetic environments. The review focused on current commercial and military applications with emphasis on both hardware technologies and software designs and methods.

The second step was the top-level design of the Synthetic Battlebridge System incorporating many of the features identified in the literature review. The initial algorithm for the Synthetic Battlebridge System is given in Figure 1. Because of the real-time nature of the Synthetic Battlebridge System, the Design Approach for Real Time Systems (DARTS) design method (Gomaa, 1984) was selected. The detailed DARTS development of the Synthetic Battlebridge System is contained in chapter 3. Figure 2 shows the initial external interfaces to the Synthetic Battlebridge System.

```
SYNTHETIC BATTLEBRIDGE SYSTEM

initialize system
while not done
    get glove gesture
    process glove gesture
    get user position
    compute eye point
    get world objects
    while world objects
        if object in field of view
            process object
            render object
        get next object
    get voice commands
    process voice commands
exit system
```

Figure 1. Synthetic Battlebridge System Initial Algorithm

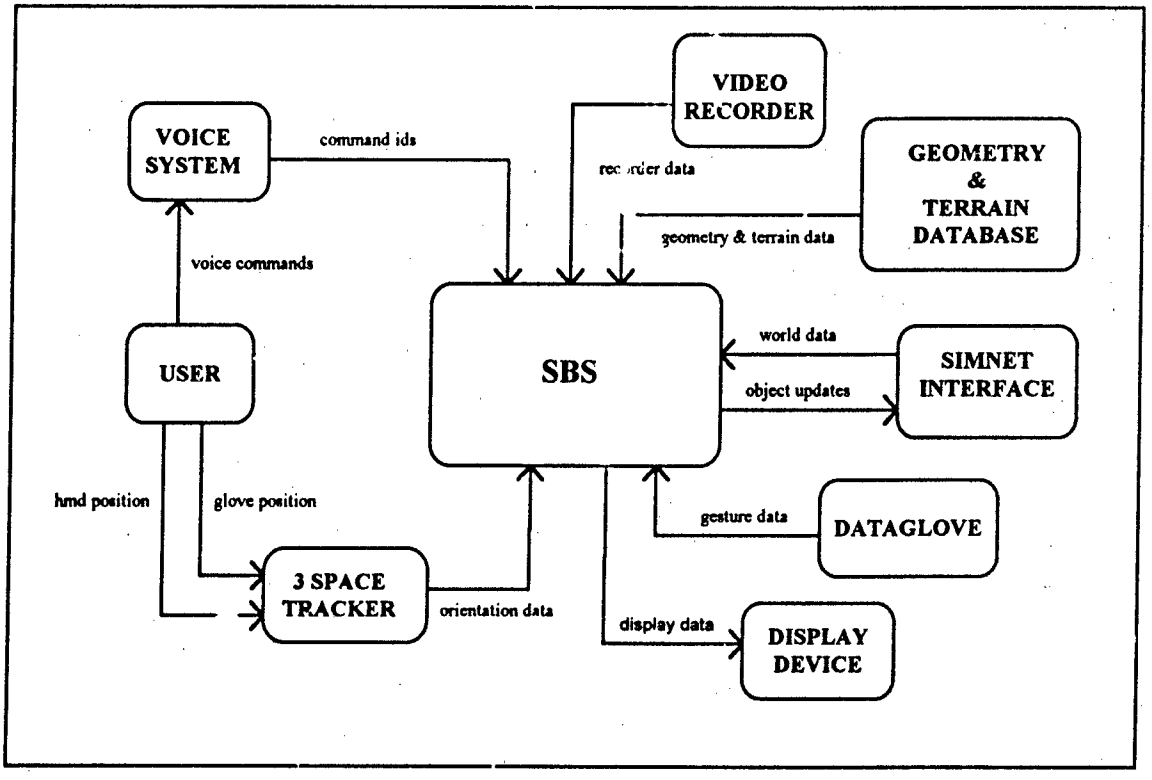

Figure 2. Synthetic Battlebridge System Initial Configuration

After the detailed design was complete the next step was to evaluate what, if any, existing code could be reused or modified for use with the Synthetic Battlebridge System. The key concerns in this evaluation was the object-oriented nature of the code, the class structures themselves, and the interfaces of the access methods.

The fourth and final step was the development of all additional supporting class structures based on Booch's object-oriented development method (Booch, 1987). Each object class was independently tested and gradually integrated into the system prototype.

1.6 Materials and Equipment

There are many issues to address in designing and implementing a synthetic environment however, the most critical performance issue is the overall system frame rate. Areas most effecting the overall frame rate include:

- Mathematical computations
- Graphics rendering pipeline
- Number of polygons
- Shading algorithm
- Texturing
- Input/output with external devices

Frame rates of less than ten frames/second look like a series of pictures rather than an animated scene and induce a noticeable lag between the user's movement and system response (Bryson, 1992). To achieve an acceptable frame rate, a special purpose, parallel graphics workstation was chosen to implement the Synthetic Battlebridge System. The Silicon Graphics IRIS 4D/440VGXT was the most cost effective workstation available. It provided both a hardware graphics pipeline and sufficient RS-232 ports for the external devices required.

All software was written using the AT&T C++ version 2.1 compiler and was designed with portability and flexibility as primary considerations. Because the AT&T C++ translator is extremely restrictive in that it does not allow many of the non-standard options available with other compilers, code compiled with it should compile with only modest changes on other systems or with other C++ compilers.

1.7 Thesis Organization

The remainder of this document describes the steps taken to create the Synthetic Battlebridge System. The next chapter describes various synthetic environment interface devices and techniques, including research conducted at AFIT. Chapter 3 describes the

design approach taken to create the Synthetic Battlebridge System, with a discussion of how the software and hardware work together to allow a user to enter and operate within the synthetic environment. Results and conclusions are discussed in Chapter 4. Appendix A provides a brief description of each software class and associated methods. A detailed user's manual for the Synthetic Battlebridge System is contained in Appendix B.

II. Synthetic Environment Technologies

2.1 Introduction

Immersive synthetic environments integrate computer graphics with various input/output and display technologies to create the illusion of immersion in a computer generated reality (Bryson, 1992: 1.2). A user interacts with this generated world containing seemingly real, three-dimensional objects in three-dimensional space.

To obtain the illusion of immersion, the user must be as unobtrusive and natural to the computer-generated environment as possible. This type of immersive environment supports the computer-human interface paradigm: to interact with a computer-generated object, as opposed to a computer-generated image of an object. The methods, techniques and technologies used in producing this illusion are the topic of this chapter. Techniques discussed are based on current commercially available technologies.

2.2 Immersive Synthetic Environments

All synthetic environment systems are composed of four basic components:

- **Image Generator**: Maintains the state of the environment, renders the environment, and interfaces to all input/output devices.
- **Tracking System**: Monitors and determines the actions and positions of the user.
- **Input Devices**: Provide user interaction.
- **Display Device**: Displays the computer generated images.

More elaborate systems may contain additional components to support voice output, voice recognition, monaural and spatially localized sound and other types of interactive devices. The primary challenge of synthetic environment development is the integration of these technologies into a single system (Bryson, 1992: 1.5). A typical system containing these components is shown in Figure 3.

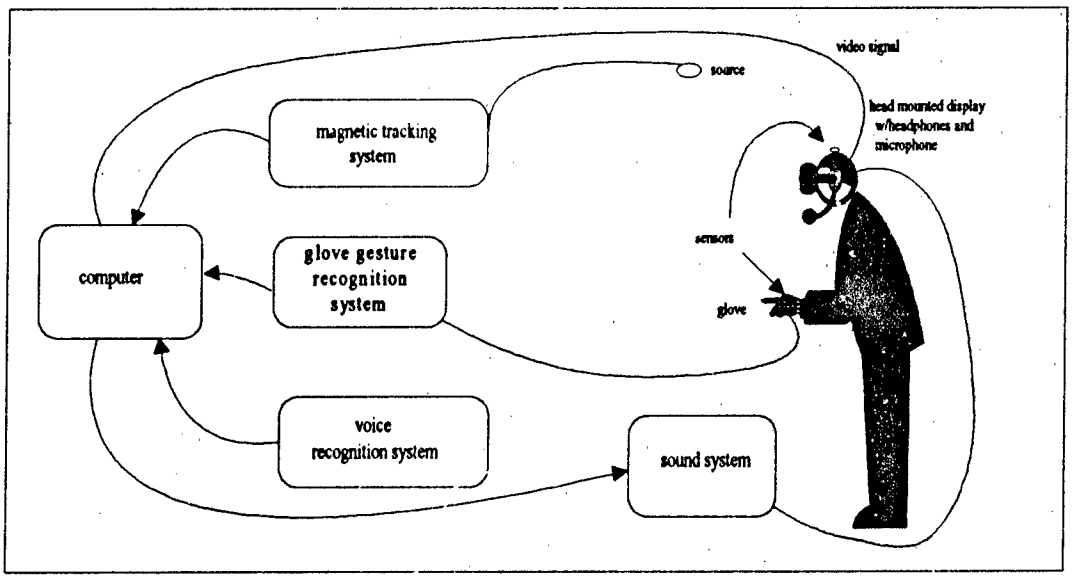

Figure 3. Synthetic Environment System

In this system, both head and hand tracking are performed by a single tracking system. The video signal supplied to the head-mounted display can be either mono or stereo. The sound system can provide the user with either monaural or spatially localized sound. The user can interact with the computer-generated environment through the glove and the voice recognition system.

An alternative configuration is shown in Figure 4. In this system the head-mounted display is replaced by boom-mounted display system that provides its own positioning information instead of tracking the user's head.

To meet the goal of an immersive synthetic environment, the system must provide a display that gives the objects a visual sense of presence and endows objects generated in this environment with the ability to interact with a user. Bryson contends certain features of synthetic environments enhance this sense of presence more than others (1992: 1.6). He suggests that there are *degrees of immersion*. Figure 5 lists the degrees of immersion he contends are provided by various display features.

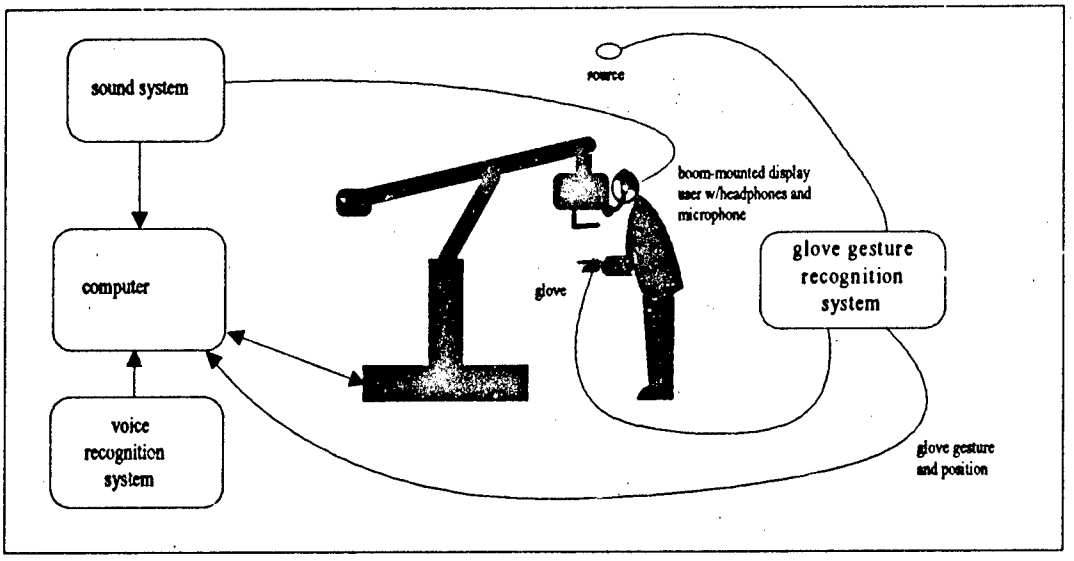

Figure 4. Boom-mounted Synthetic Environment System

DISPLAY FEATURES	DEGREES OF IMMERSION
monochrome	not very immersive
color	↓
high resolution	
stereo	
head tracking	
wide field of view	
head mounted	very immersive

Figure 5. Immersive Display Features (Bryson, 1992: 1.6)

Each of these features is independent and additive. For example, adding stereo to a head-mounted display would increase the user's sense of immersion more than adding color to the head-mounted display. The degree of immersion is also affected by the interaction devices as shown in Figure 6.

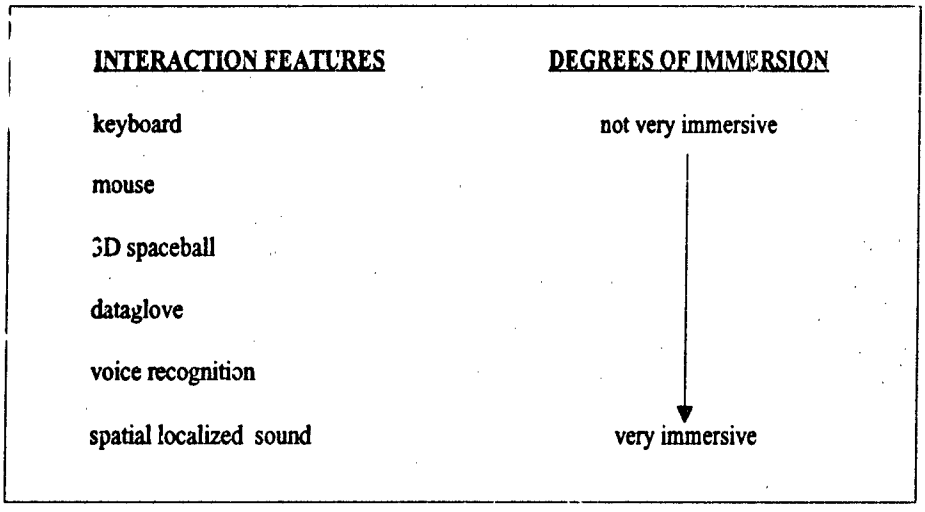

Figure 6. Immersive Interaction Features (Bryson, 1992: 1.7)

2.3 Tracking Systems

As previously stated, the goal of any synthetic environment is to make the user part of the environment. To accomplish this, all interactions with objects in the environment should mimic, as much as possible, the interactions a user would have with real objects. By tracking parts of the user's body, fulfillment of this goal can be approached. For example, by sensing the position and orientation of the user's head, the synthetic environment can be displayed from the user's perspective. By tracking the user's fingers, gestures can be determined. The gesture informs the system what the user wants to happen and the position of the hand indicates where.

Current tracking systems fall into one of two classes:

- **Position tracking**: provide location and orientation in three-dimensional space.
- **Angle measurement**: detect the bend angle of some body part.

Four commonly used position tracking technologies are electromagnetic, mechanical linkage, ultrasonic, and optical. Each of these technologies has advantages and disadvantages when incorporated into a synthetic environment.

2.3.1 Electromagnetic Tracking. Electromagnetic trackers use magnetic fields to determine the position of a sensor relative to a fixed source. Two companies, Polhemus (Polhemus, 1987) and Ascension Technologies (Ascension, 1990), use the same basic design principle in their tracking systems. A set of three orthogonal coils sequentially pulsed acts as a signal source. Each pulse transmits an electromagnetic signal that is detected by a sensor. The sensor also contains three orthogonal coils, which measure the signal strength of the pulse in three directions. These measurements are used to determine the location and orientation of the sensor.

Electromagnetic trackers have the advantages of relatively low cost, increased mobility, and allow tracking in an arbitrary position and orientation in a given volume of space (Ferrin, 1991). However, they are notorious for their decreasing accuracy at distance and their susceptiblity to interference from ferrous materials. They also suffer from significant time delays due to considerable internal signal filtering computations.

2.3.2 Mechanically Linked Tracking. Mechanical linkage systems use a jointed structure arranged to allow a free end of the structure to be moved to an arbitrary position and orientation. Sensors at the joints detect the angle of the joint and a simple concatenation of translates and rotates determines the position and orientation of the free end about a fixed base. An example of mechanical linkage is shown in Figure 7.

The original design is credited to Scott Fisher and Jim Humphries of the NASA Ames Research Center VIEW Lab and is used by Fake Space Labs in their BOOM™ display system (McDowall and others, 1990).

Mechanical linkage has the advantage of being extremely accurate and responsive. The main disadvantages are encumbered movement and high cost.

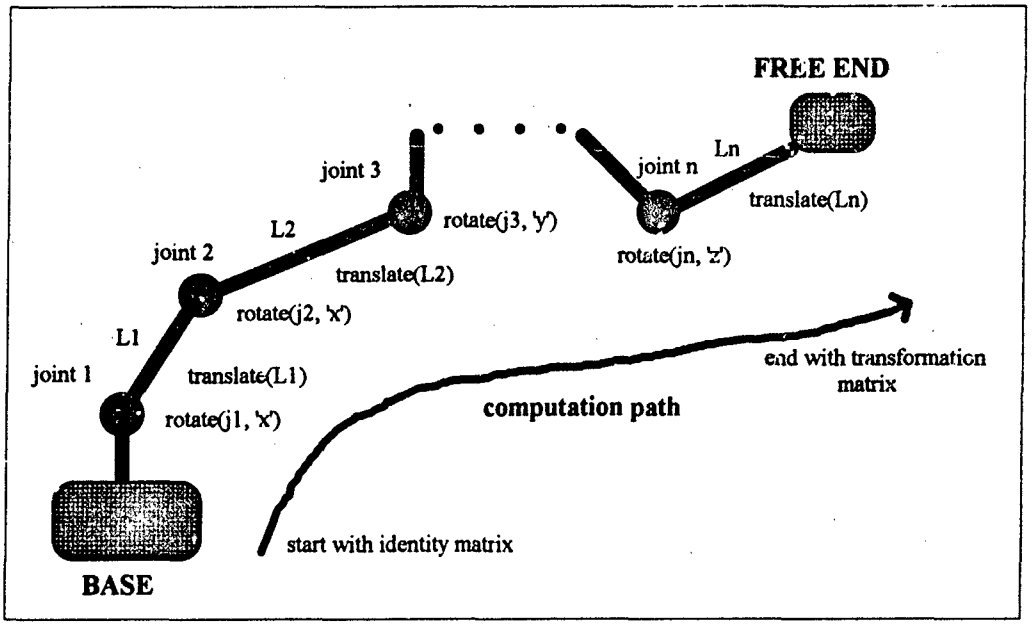

Figure 7. Mechanical Linkage (Bryson, 1992: 1.11)

2.3.3 Ultrasonic Tracking. Ultrasonic tracking is based on the travel time of a sound signal. The time it takes for a signal to travel to a source consisting of three receivers is used to calculate the position of a transmitting sensor. If the source contains three transmitters, the orientation of the sensor can be determined. Figure 8 shows a typical ultrasonic tracking setup.

Ultrasonic tracking is inexpensive and easy to build. One device commercially available using ultrasonic tracking is the Mattell Powerglove™. Logitech is also using ultrasonic head tracking with their stereo CRT screens (Axt, 1987).

The disadvantages of ultrasonic tracking are the inaccuracy of the tracked signal and the necessity of maintaining a clear line of sight between the transmitter and the receiver.

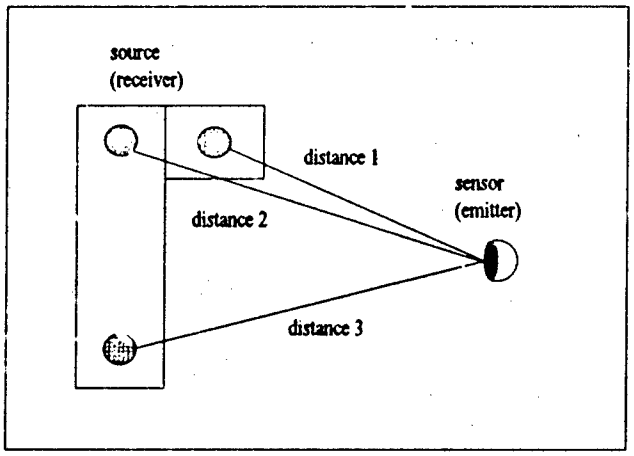

Figure 8. Ultrasonic Tracking System (Bryson , 1992: 1.12)

2.3.4 Optical Tracking. Optical tracking systems use light sources and photosensitive receivers to measure orientation and position data. The sources transmit light in predetermined patterns that identify their locations. On the basis of the pattern received by the sensor and the time from the firing of the light, the position and orientation of the sensor can be determined (Ward and others, 1992).

The primary advantage of optical tracking is its high degree of accuracy. The disadvantages include high cost and the potential occluding of the light signal from the sensor.

2.4 Angle Measurements

Another way to track parts of a user's body is by measuring joint angles. The angles can be used to determine the position of various body parts. Angle measurement systems are typically used in hand gesture recognition systems like the VPL DataGlove™.

The VPL DataGlove™ uses specially treated optical fiber at several finger joints to measure the bend of the fingers (VPL, 1989). This technology has also been applied to measure other body joints in the VPL DataSuit™. Another measuring system, used in the Exos Dexterous Handmaster™, is a jointed exoskeleton attached to each finger segment

with potentiometers at each joint. One other company, Virtex Technologies, uses a electromagnetic strain sensor vice optical fibers to measure angle bends.

All gloves require calibration for each user and in some cases recalibration by the same user (Pond, 1992). Calibration is usually performed by measuring two angles per joint. Gestures are then recognized when the measured angles match the stored calibrated data. The primary difficulty with gesture recognition is that users do not always make the same gesture the same way.

2.5 Tracking Data

The data returned by these trackers is a set of numbers representing position and orientation in three-dimensional space. Most of the systems offer a choice of two or more forms and formats.

Position data typically consists of a vector of three numbers in a coordinate system centered at the source. Usage of these numbers is fairly straightforward.

Orientation data is much more complex and most tracking systems offer two or more ways of representing this data. The Polhemus systems offer Euler angles, direction cosines, and quaternions (Polhemus, 1987). The Ascension systems offer Euler angles, direction cosines, and a 3 X 3 rotation matrix (Ascension, 1990). The Fake Space Labs' BOOM™ systems provide Euler angles and a 3 X 3 rotation matrix (Fake Space Labs, 1992).

Euler angles are the concatenated angles of azimuth (yaw), elevation (pitch), and roll about the origin (Figure 9). The advantage of euler angles is that they require only three numbers or six bytes and are well suited to most graphic pipelines.

Direction cosines are the cosines of the angles between the sensor's X, Y, Z axes and the X, Y, Z axes of the source. A 3 X 3 (attitude) matrix containing the direction cosines for the Polhemus 3SPACE ISOTRAK™ system is shown in Figure 10. The

direction cosines of the sensor's X-axis are in column one, the sensor's Y-axis are in column two, and the sensor's Z-axis are in column three.

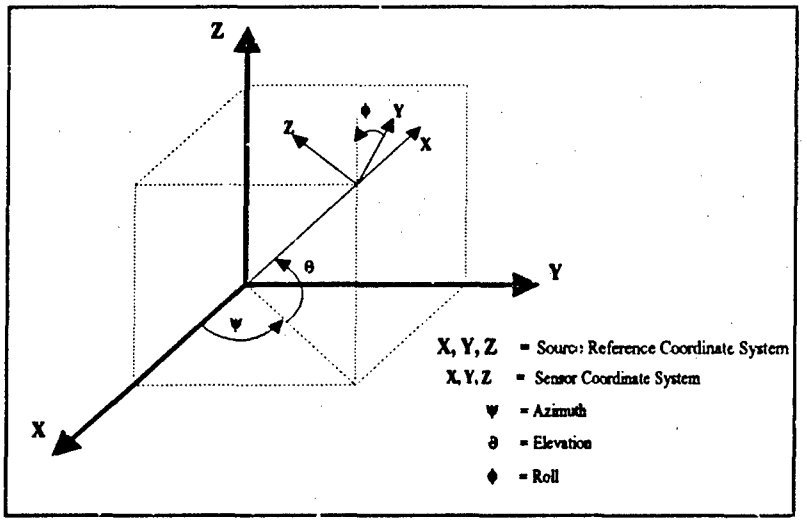

Figure 9. Euler Angles (Polhemus, 1987: B-5)

$$\begin{bmatrix} CA * CE & CA * SE * SR - SA * CR & CA * SE * CR + SA * SR \\ SA * CE & CA * CE + SA * SE * SR & SA * SE * CA - CR * SR \\ -SE & CE * SR & CE * CR \end{bmatrix}$$

where
SA = Sine Azimuth CA = Cosine Azimuth
SE = Sine Elevation CE = Cosine Elevation
SR = Sine Roll CR = Cosine Roll

Figure 10. Direction Cosines (Polhemus, 1987: B1-B2)

Quaternions are a four parameter quantity representing a vector and a scalar of the following form:

$$q = w + xi + yj + zk$$

Quaternions can be used to represent a sensor's orientation without the need for trigonometric functions. The x, y, and z values of the unit quaternions ($w^2 + x^2 + y^2 + z^2$

= 1) specify an axis and w specifies a rotation about that axis by the angle $2\cos^{-1}(w)$ (Shoemake, 1985). The quaternion can be converted to a rotation matrix by using the following matrix:

$1 - 2y^2 - 2z^2$	$2xy + 2wz$	$2xz - 2wy$
$2xy - 2wz$	$1 - 2x^2 - 2z^2$	$2yz + 2wx$
$2xz + 2wy$	$2yz - 2wx$	$1 - 2x^2 - 2y^2$

2.6 Visual Displays

Several display technologies are available for use with immersive synthetic environments. Following are some of these in order of increasing immersion:

- conventional workstation screens
- conventional workstation screens with stereo
- wide screen with stereo
- multiple wide screens with stereo
- head tracked, multiple wide screens with stereo
- head mounted, head tracked wide field stereo

Conventional workstations screens with stereo glasses are becoming common with research in developing stereo systems without the need for special glasses (Bryson, 1992: 1.16). Widening the screen enhances the user's sense of immersion, with multiple wide screens serving to widen the field of view even more. Adding head tracking enhances the presence of objects in the environment.

The most immersive technology is a head-mounted display. The display does not actually have to be physically mounted on the user's head as in the Fake Space Labs' BOOM™ systems. Stereoscopic vision is obtained by using two screens with slightly different images. An extended analysis of head-mounted optics can be found in an article by Robinett and Rolland (Robinett and Rolland, 1992). Head tracking allows rendering of objects in the environment from the user's point of view.

Current head-mounted displays are based on three different technologies: liquid crystal displays (LCDs), miniature cathode ray tubes (CRTs), and fiber-optic cables. LCDs have the advantage of being flat and extremely lightweight. They suffer from low resolution and limited brightness. CRTs have a distinct advantage over LCDs in image quality, but the additional optics used to give a wide field of view make them too heavy for head-mounted systems. This weight problem can be solved by supporting the CRTs on a counterweighted assembly. There are a number of commercially available head-mounted LCD displays with VPL's EyePhones™ the best known. LEEP Systems and Virtual Research, among others, produce head-mounted displays similar to VPL's.

A popular CRT based system is the Fake Space Labs' BOOM™ device. The BOOM™ systems support two CRTs in a head assembly with a counterweighted yoke (Figure 11). The assembly is jointed to allow six degrees of freedom, although roll is extremely difficult.

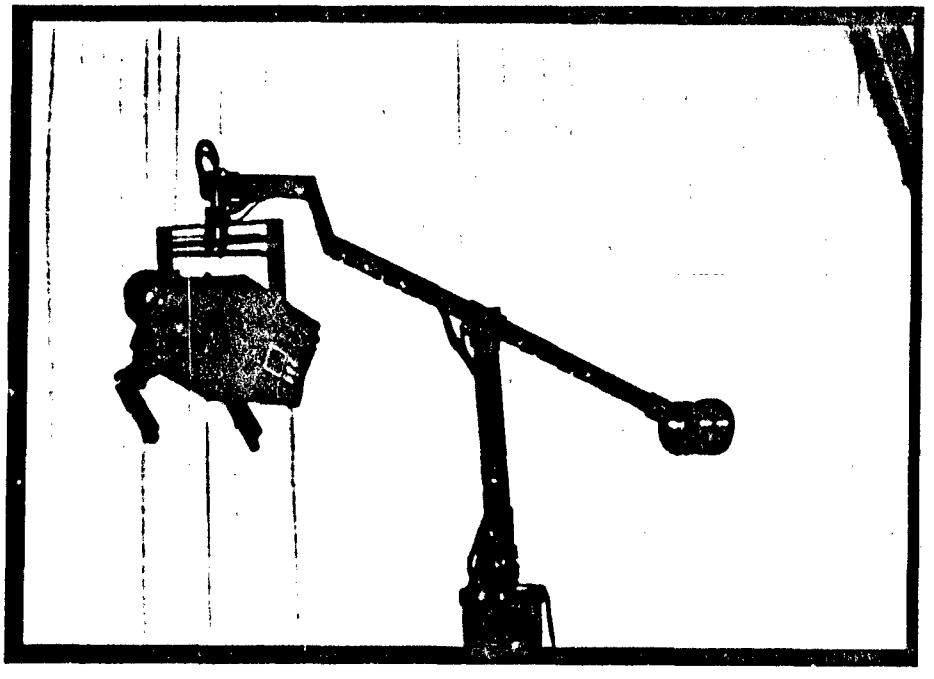

Figure 11. Fake Space Labs BOOM2M™

The BOOM™ is held up to the user's face by one or two hands with the initial impressions of most users being similar to that of using a periscope. The jointed assembly is mechanically linked to the BOOM's base and can be used for accurate tracking.

A head-mounted display system under research at AFIT is the Polhemus LookingGlass™ (Figure 12). The system, designed by Bill Polhemus, uses two Sanyo Liquid Crystal Color Video Projectors projecting a national television standard code (NTSC) video signal onto the ends of fiber optic bundles. A Nikkor 35mm camera lens is used to focus the image onto the fiber bundle. Because of the high light output of the projectors, a neutral density filter is used on the camera lens. The fiber bundle is composed of a 1667×1333 matrix of 6 mircon fibers. This results in over 2.3 million fibers with an aspect ration of 5 to 4. The opposite ends of the bundles are tapered and fused onto a 25×25 mm rectangle glass. A generated image is then reflected onto a pair of half-silvered mirror glasses mounted 1.322 inches from the center of the intersurface of the glass. The generated image has a field of view of 42°×30° per eye (Figure 13) (Polhemus, 1993).

The advantages of the LookingGlass™ include high resolution, good brightness range, and the ability to adjust the display for most user vision problems. The disadvantages are encumbered movement, weight, and a "peppered" image display. This peppering is due to broken fibers and resultant dropouts in the image; broken fibers do not transmit light.

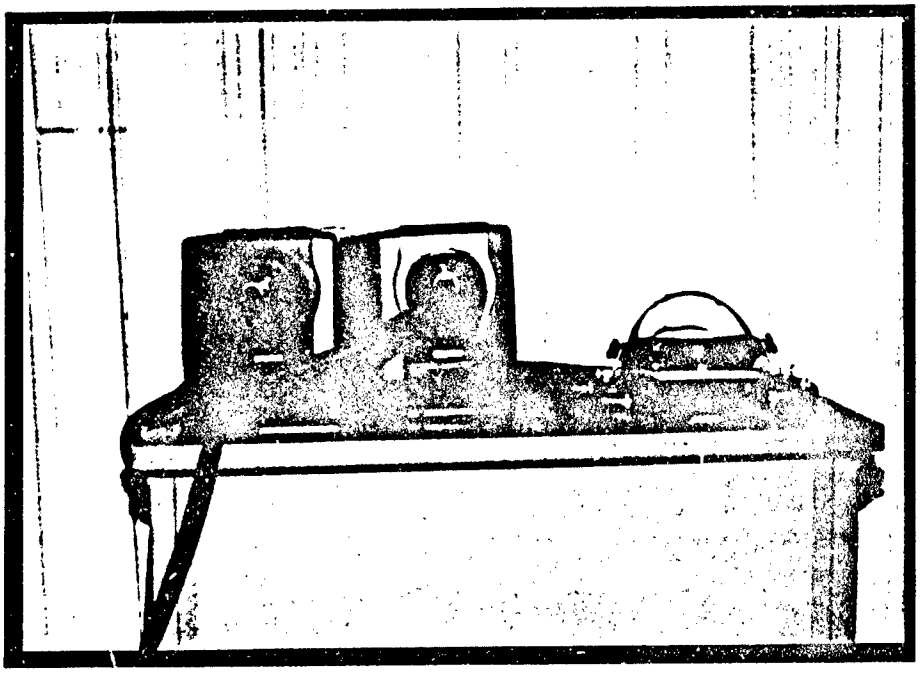

Figure 12. Polhemus LookingGlass™ System

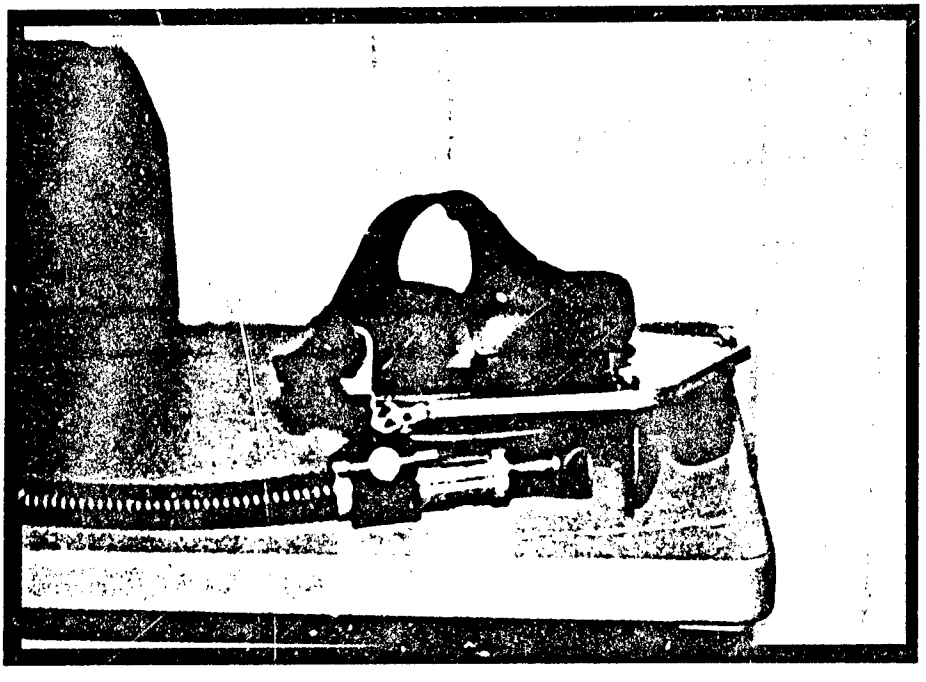

Figure 13. LookingGlass™ Head-Mounted Display

2.7 Other Input/Output Devices

The need for complete sensory input to create the illusion of reality was recognized by Ivan Sutherland in 1965 (Sutherland, 1965). Tracked head-mounted displays provide the greatest level of immersion of all the display technologies, but these systems cannot provide more than a visual interface to the synthetic environment; other interface devices are necessary to deepen the immersion. Haptic feedback, force/torque controllers, sound systems, and voice recognition systems have been used to enhance the immersion and provide additional user interface to synthetic environments.

2.7.1 Haptic Systems. Haptic systems allow a user to interface with the environment through the sense of feel. These systems generally provide feedback to a user through one of the following methods:

- **Mechanical Linkage**. Mechanical actuators are used to convey force and texture data to a user.
- **Pneumatics**. Pneumatic systems convey pressure levels to a user with inflatable air bladders.

2.7.2 Force/Torque Controllers. Force/torque controllers use sensors to measure the force and torque in three dimensions relative to a center point inside the handle of the controller (Figure 14). Force/torque controllers can be used as six degrees of freedom joysticks with the most notable being the Spatial Systems Spaceball.

2.7.3 Sound Systems. Sound can be used to indicate events and provide feedback to the user. Sound can be provided in either monaural or spatially localized form. Monaural sound can be used to indicate various states of the synthetic environment, with different sounds used to indicate different states.

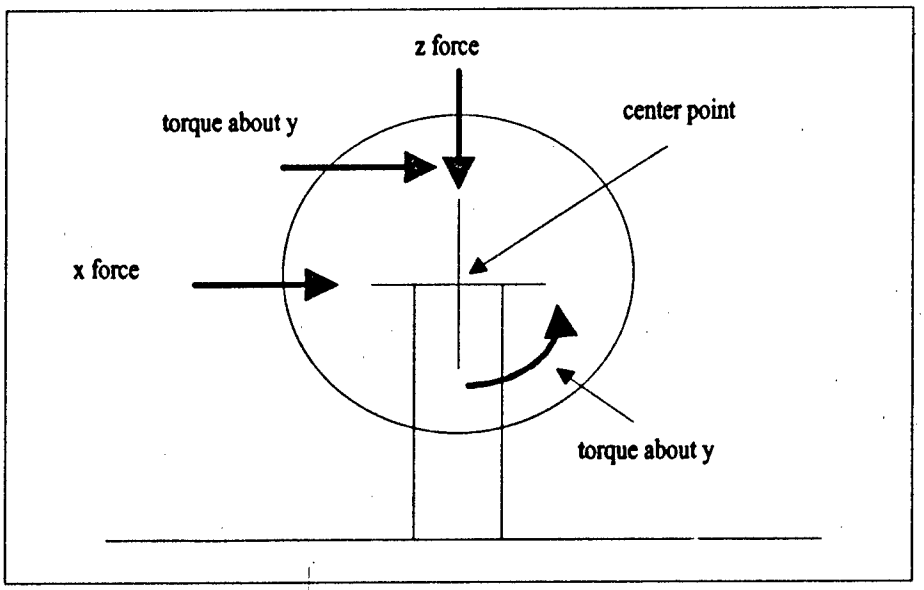

Figure 14. Force/torque Controller (Bryson, 1992: 1.25)

Spatially localized sound can provide a user with a significant increase in the sense of presence. Simple stereo sound in a synthetic environment enhances the immersion but does not give true spatially localized sound. Scott Foster of Crystal River Engineering developed the convolvotron, a device that gives sound a true sense of three-dimensional location. Because the sound has this three-dimensional quality, it can be used to assign qualities to static and dynamic objects in the synthetic environment (Scarborough, 1992).

2.7.4 Voice Recognition Systems. Conventional voice recognition systems are based on either pattern matching techniques or stochastic modeling. Pattern based recognizers may use words, phonemes, or other features characteristic of speech signals as the input pattern. The input pattern is identified by finding the closest match between this pattern and all the stored patterns in the vocabulary file of the recognizer (Recla, 1989).

The use of the Markov model to stochastically model speech has gained favor in recent years. Recognition accuracies are similar to pattern based systems with additional improvements anticipated as research continues. This type of voice recognition builds a

Markov model for each word in its vocabulary. An input word matches a vocabulary word whose Markov model has the highest probability of matching the Markov model given to the input word.

Voice recognition systems have the advantage that they are unobtrusive in a synthetic environment and, if properly designed, voice commands offer a more intuitive interface than using hand gestures.

The disadvantages of using voice recognition systems are the variability of speech (both within and across speakers) and the calibration required for each user.

2.8 Integrated Synthetic Environments

According to Bryson, the integration of the various parts of a synthetic environment into a coherent working system is the most challenging and difficult aspect of synthetic environment development:

> The virtual environment designer is constantly struggling with conflicts like that between accuracy of computation and speed of performance, or between color displays and resolution. Thus the design of an integrated virtual environment system often involves compromises between conflicting aspects of the system. Most of these compromises are to work around deficiencies in the technology, and so will change with time. The intelligent choice of these compromises is what makes or breaks a virtual environment application. (1992:1.26)

Several companies are providing products to assist in the construction of synthetic environments. VPL Research has both hardware and software products available including a complete system. VPL's Body Electric™ is a data flow language designed to assist in low level development. Sense8's World Toolkit™ is a library of C routines for low level development.

Each synthetic environment application poses different interface problems with potentially different solutions. Regardless of the application, there are several interface issues common to all synthetic environments:

- impact of performance on system usability
- integration of several coordinate systems
- navigation within the synthetic environment
- organization and integration of data
- design and use of interactive tools.

Two noteworthy applications that have successfully addressed these issues are the Virtual Windtunnel (Levit and Bryson, 1992) and the Virtual Planetary Exploration project (Hitchner, 1992).

The Virtual Windtunnel is designed to visualize numerically simulated aerodynamic flows. The system consists of a Silicon Graphics 380 VGX workstation, Fake Space Labs BOOM2M™ and BOOM2C™, a VPL DataGlove™ with a Polhemus 3SPACE tracker, a mouse, and a keyboard. The use of the BOOM™ displays allows a user to engage/disengage from the environment as quickly as possible. The DataGlove™ is used as the primary input device with limited gestures for adding, deleting and moving groups of visualization tools (Levit and Bryson, 1992:4.4).

The Virtual Planetary Exploration (VPE) is a research tool for investigating concepts, methods, and interaction techniques for the design of planetary exploration workstations. The system consists of a Stardent GS2000 graphics workstation, a Virtual Research Flight Helmet and a VPL Model I EyePhone™, a Polhemus 3SPACE ISOTRAK™ tracker, and a Spatial Systems Spaceball. The initial application of the VPE was the visualization of the surface of Mars form data collected by NASA's Viking orbiter (Hitchner, 1992:6.2).

2.8 Conclusion

This chapter discussed some of the methods, techniques and technologies used in developing immersive synthetic environments. The following chapter describes the application and implementation of many of these areas in the design and development of the Synthetic Battlebridge System.

III. System Design and Implementation

3.1 Introduction

Real-time systems frequently consist of several tasks executing concurrently on multiple processors. Each task executes in a sequential manner and in many instances, must communicate and synchronize with other tasks (Levi and others, 1990). The design and development of these systems has usually been done as an extension of single task design methods.

This chapter presents an evaluation of current design method as they apply to real-time systems, an extended description of the Synthetic Battlebridge System design process, and detailed descriptions of the Synthetic Battlebridge System's software, environment, and operation.

3.2 Real-Time Design Methods

The first step in the design of the Synthetic Battlebridge System was to evaluate several current design methods as they relate to real-time systems. Only some of the more popular and accepted design methods were evaluated. These were Structured Programming, Structured Design, Functional Decomposition, and Mascot. For a more detailed evaluation of these and other design methods see (Gomaa, 1979).

Structured Programming methods first appeared in the 1970's. The original basis for structured programs was the concept that the logic in all computer programs could be broken down into three basic forms, sequential, conditional, and iterative.

The three basic principles of Structured Programming are abstraction, modularity, and stepwise refinement. Abstraction is the concept that the program design should begin with an overview of what should be done, not how to do it. If possible, the problem should be expressed in spoken English without technical jargon. Modularity refers to the structure of the code. The program should be broken into a series of modules. When modules are designed, they should be structured with high cohesion, and low coupling. A

module with high cohesion performs a single, well-defined task. A module with low coupling does not affect the performance or results of other modules when it is changed. Stepwise refinement is probably the best known concept of structured programming. The concept of stepwise refinement is simple, keep breaking down the problem into smaller and smaller problems or tasks, until each problem statement describes an atomic task. That is, break down the problem to its component parts, and concentrate on each small problem as opposed to the problem in large.

Structured Programming methods are control-structure-oriented methods primarily applicable to detailed program design and implementation. They do not handle the problem of how to decompose a system into concurrent tasks, and consequently are not appropriate for the Synthetic Battlebridge System.

Structured Design methods deal with how to decompose systems into modules. In principle, they are a collection of guidelines and techniques to help the designer choose between good and poor design techniques, at the modular level (Yourdon, 1976). Structured Design introduced techniques for documentation throughout the life cycle, evaluation criteria, heuristics for program evaluation, design strategies, and implementation strategies. The modular design was a needed step in the growth of software development. Structured Design can lead to highly functional modular designs but provides no help with structuring a system into tasks, nor does Structured Design address the issue of internal module design (Gomaa, 1984).

Functional Decomposition is an extension to Structured Design methods that decomposes a system into modules. It provides additional guidelines and techniques for breaking down large programming projects into the proper modules for development. Again, this was a needed evolution of program design, but in real-time applications, functional decomposition fails to address the issue of decomposing a system into concurrent tasks.

The Mascot method deals specifically with structuring a system into processes or

tasks and defining the interfaces between them. However, it does not address how to structure a system into tasks nor how the internal structure of the tasks are designed.

3.3 Synthetic Battlebridge System Design

All the above design methods have limitations when used in the design of real time systems. Each method contained certain aspects essential to the design of the Synthetic Battlebridge System but, what was needed was a method that incorporated all the requirements of the Synthetic Battlebridge System.

In real-time systems, the system design defines how the system is decomposed into tasks. The design method I used during the design of the Synthetic Battlebridge System was DARTS. DARTS is a software design method that can be used to structure a distributed real-time application into subsystems that consist of concurrent tasks (Gomaa, 1989). DARTS extends the Structured Analysis/Structured Design methods by providing a set of criteria for structuring an application into subsystems and an additional set of criteria for structuring the subsystems into concurrent tasks with a means to define the interfaces between these tasks. Tasks may communicate by means of messages and information hiding modules that support concurrent access to data stores.

The DARTS design method starts with black-box requirements specifications that define what features the system will provide without regard to how they will be provided. Like all Structured Design/Structured Analysis methods, DARTS is intended to be an iterative process and consists of the following steps:

- Perform data flow analysis using structured analysis.
- Structure system into subsystems.
- Define interfaces between subsystems.
- Structure each subsystem into concurrent tasks.
- Define interfaces between tasks.
- Design each task using structured design.

3.3.1 Data Flow Analysis. Starting with the functional requirements of the Synthetic Battlebridge System, which defined the system features without consideration of how the features would be provided. the data flow through the system was analyzed, and the major functions were determined. A system context diagram (Figure 15) and a data flow diagram (Figure 16) were developed and decomposed to identify major subsystems and their major components. The data flow diagram contains transform bubbles representing functions carried out by the system and arrows representing data flows between transforms.

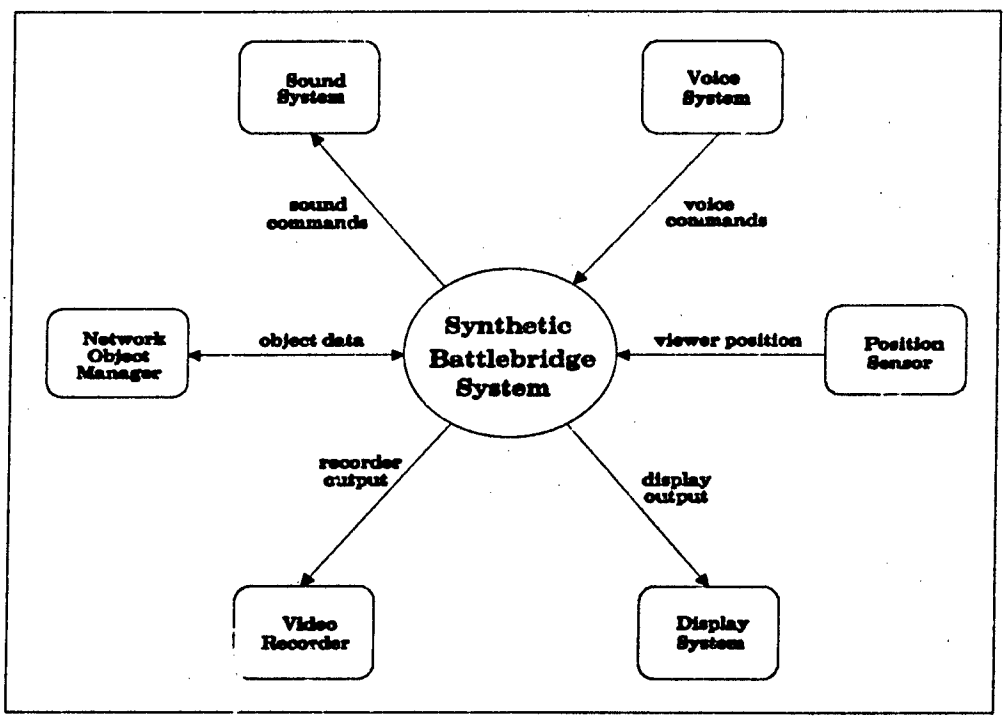

Figure 15. System Context Diagram

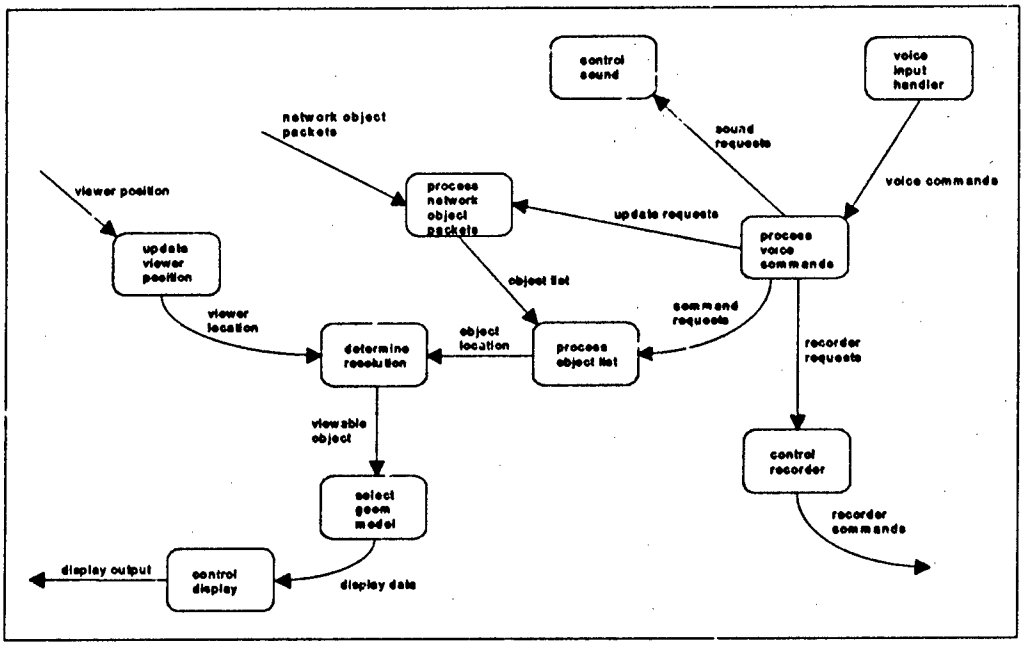

Figure 16. System Data Flow Diagram

3.3.2 Task Concurrency. Once all the functions in the Synthetic Battlebridge System and the data flows between them had been identified, the next step in the DARTS method was to determine how the system could be structured into concurrent tasks. The main consideration was the asynchronous nature of the tasks within the subsystems. It was critical to determine which tasks or group of tasks could function independently. The DARTS criteria for determining whether a task should be separated or grouped with other tasks are the following:

• Dependency on I/O. Tasks connected to I/O devices are constrained by the speed of the I/O device and need to be a separate task.

• Time-critical functions. These functions need to run at a high priority and therefore are separate tasks.

• Computational requirements. Computationally intensive functions can run at lower priorities.

• Functional cohesion. Tasks that perform a set of closely related functions can be grouped together.

• Temporal cohesion. Tasks that perform functions that are carried out at the same time can be grouped so that they are executed each time the task receives a stimulus.

• Periodic execution. Tasks that need to be executed periodically need to be structured as separate tasks.

For the Synthetic Battlebridge System the Display Manager, the Command Processor, the Sound Controller, and the Recorder Controller are all I/O dependent and need to be structured as separate tasks (Figure 17). The Network Manager task is formed by both the I/O dependency of reading the network and the time-criticality of processing this information. The Viewer Position Input Handler is composed of the I/O dependent tasks of reading the position sensor and mouse button and the functional cohesiveness of processing this data. The remaining tasks are grouped into a single task, the Object Controller, due to their functional cohesion.

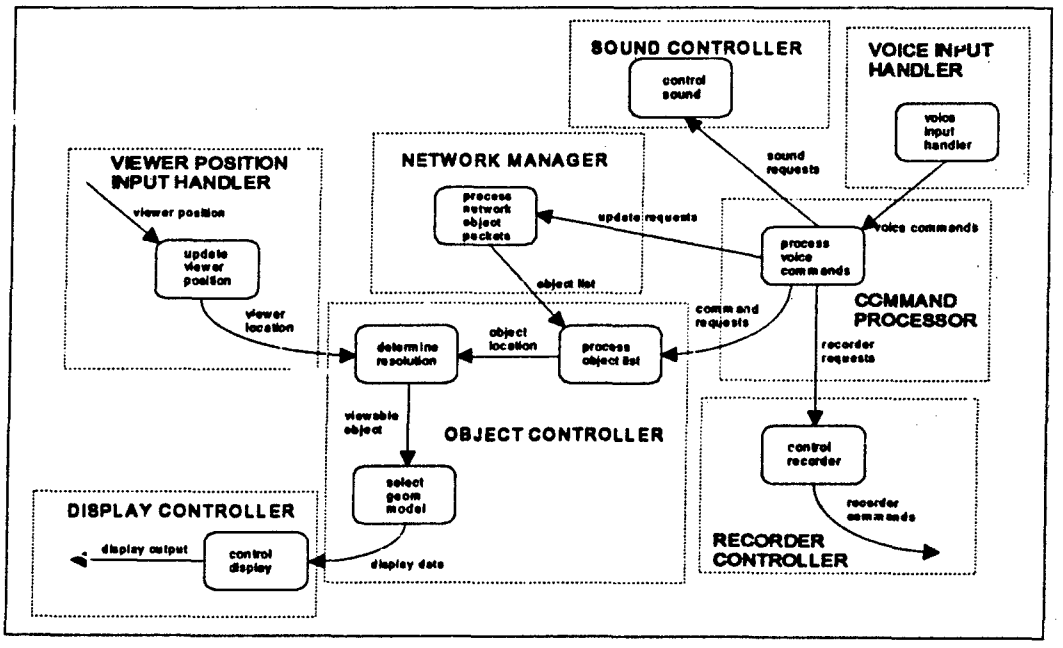

Figure 17. Task Identification

After all the tasks had been identified, the next step was to define the interfaces between the tasks. DARTS has two classes of task interface modules, task communication modules (TCM) and task synchronization modules (TSM). The TCM handle all communications between tasks and define the data structures and access procedures to these structures. There are two different types of TCMs, message communication modules (MCM) and information hiding modules (IHM). MCMs are used for communications between tasks, while IHMs are used to share data between tasks. The interfaces for the Synthetic Battlebridge System are shown in Figure 18.

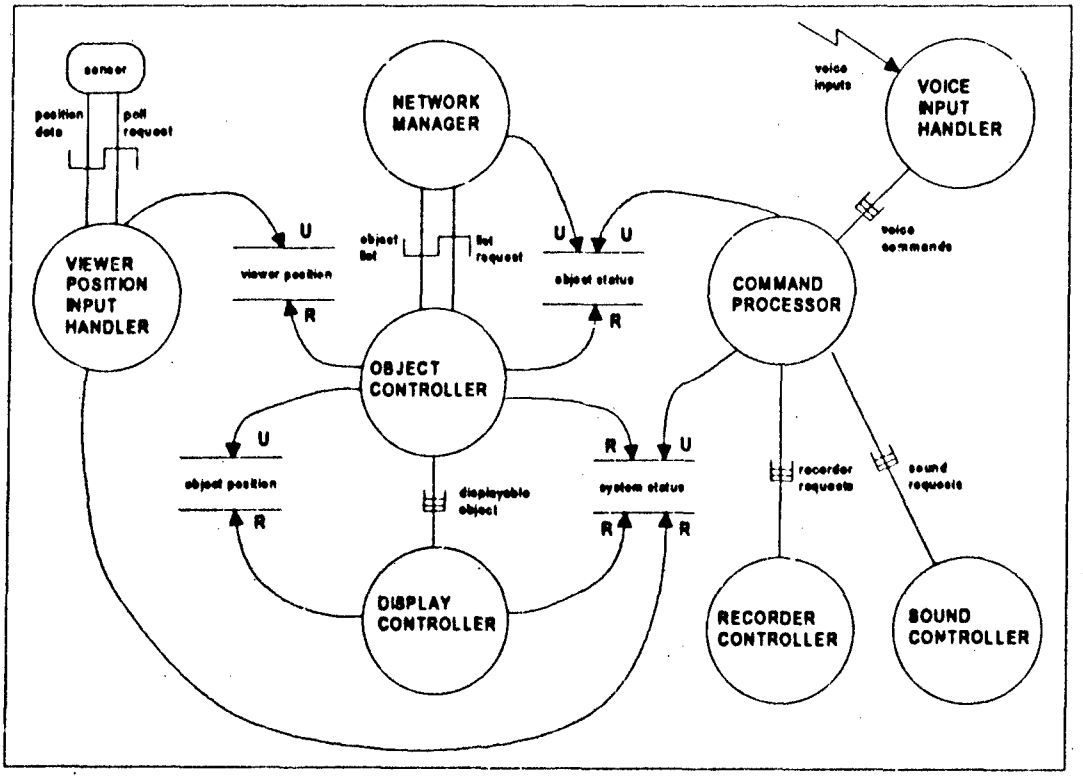

Figure 18. Task Interfaces

Voice commands are queued up for the Command Processor by the Voice Input Handler. Therefore, the interface between them consists of a message queue. The same is

also true of the interface between the Command Processor and the Sound Controller as well as the interface between the Command Processor and the Recorder Controller.

The Viewer Position Input Handler sends a poll request to the sensor and waits for a data packet containing position and orientation data. The position data is validated and based on certain status flags the viewer position is updated. This communication is an example of a closely coupled message communication. An MCM is used to provide this closely coupled communication mechanism. An MCM is also used to provide the tightly coupled communication between the Network Manager and the Object Controller.

The Object Controller processes the objects from the database based on active status flags and the position of the object with respect to the viewer. It outputs active objects in the field of view to the Display Manager.

The object status, system status, viewer position, and object tracks data stores are used to store current system information. The object status store is updated by both the Network Manager and the Command Processor. The Object Controller reads the object status to determine what objects are active and if they should be displayed. The system status store is updated by the Command Processor and read by the Object Controller, the Display Manager, and the Viewer Position Input Handler. System status flags indicate the type of objects to display, the viewer's perspective, and special status displays such as help menus and object information. The viewer position store is updated by the Viewer Position Input Handler and used by the Object Controller to determine which objects to display and at what resolution to display them. The object tracks data store contains a history of a selected object's locations. This store is updated by the Object Controller and used by the Display Manager.

Since access to the data stores is made by multiple tasks, access must be synchronized by access routines. These stores and their access routines constitute an IHM.

After the task interfaces were defined, the last step in the DARTS design method was the design of the individual tasks. Since each task at this point in the design represented an object or class of objects, I designed the individual tasks using an object-oriented development approach. This decision was supported with existing and concurrently developed object classes for the Network Manager, the Sound and Recorder Controllers, and the Viewer Input Position Handler.

3.3.3 System Development. Gomaa (Gomaa, 1986) suggests the incremental development approach to developing real-time systems because it encourages prototyping to get a working subset of the system as early as possible. Since most real-time systems deal with a large number of external events, a small subset of these can be selected to test the key interfaces of the system. This early subset is evolved into the final system.

My experience with the Synthetic Battlebridge System supported Gomaa's assertions. I found the incremental development method provided several key advantages:

- A psychological boost to both the developer and the user.
- A prototype baseline to test various parts of the system, explore feasibility of additional user requested features, and examine different equipment/performance configurations.
- A platform for taking performance/timing measurements.

Gomaa's steps for the incremental development approach are outlined as follows: After complete system and task design, detailed design of each module is effected; coding only begins after this is completed. The coding and testing however, is done in stages.

During the task design of the Synthetic Battlebridge System, the interfaces between and within object classes were fully defined so class development could proceed independently. This meant coding of some classes was underway while other classes were still being designed.

I found an event sequence diagram a useful development tool for design and incremental development. The diagram shows the sequence of actions expected to take place between tasks or task groups when an external event occurs.

The event sequence diagram (Figure 19) for the Synthetic Battlebridge System is based on the task structure chart of Figure 18. Only those tasks involved in processing the action are shown. Actions are numbered sequentially in the order they are triggered. Messages and events are shown and numbered in the order in which they occur. For example, when the user issues a record command, the voice input is received by the Voice Input Handler task, which sends a start record message to the Command Processor task. This in turn signals a start recorder event to the Recorder Controller. In addition, the recorder status is updated and subsequently read by the Display Manager to display the recording icon. The diagram provided the following benefits:

• A better understanding of the dynamic interactions of the system. The diagram serves as a design tool to aid in visualizing the task interactions.

• A guideline for deciding how to stage incremental development. By following an event sequence, the required interactions to a task could be determined.

• A platform for designing test cases. The diagram facilitates the selection of input test cases along with identifying expected output.

3.3.4 System Integration. The objective of system integration with any real-time system is to test the interfaces between tasks. The system integration plan should define the order in which the tasks are to be tested, the sequence the tests are performed in, and which tests can or will be conducted in parallel.

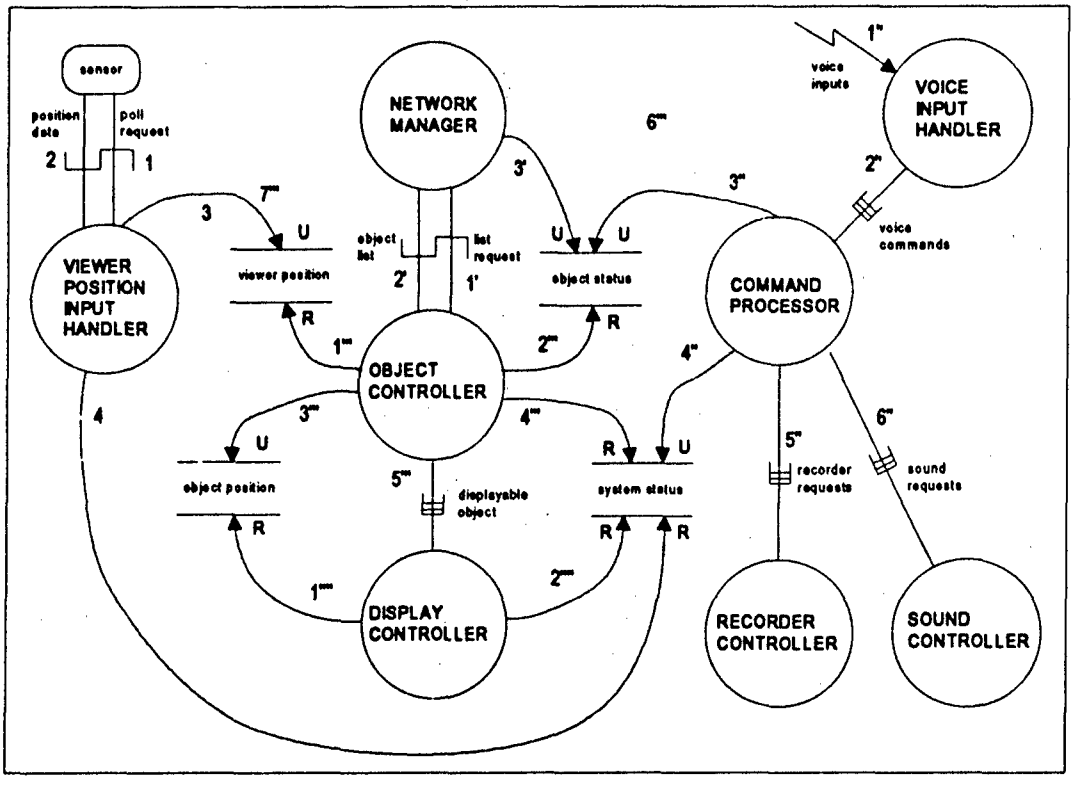

Figure 19. Event Sequence Diagram

Since communication and synchronization are critical to task interfaces in the Synthetic Battlebridge System, it was essential to test these areas early. In a TCM, the synchronization and mutual exclusion conditions necessary to ensure consistent and correct access to the data is provided by the access procedures.

A TCM of the IHM type consists of a data structure and the routines to access the data structure. Initially, an IHM can be tested using standard unit testing however, since the IHM can be accessed by more than one task it was necessary to test the synchronization aspects by having at least two tasks invoke the IHM concurrently.

A TCM of the MCM type handles communication between tasks by either loosely or closely coupled messages. MCMs are tested by setting up sending and receiving tasks that use the MCM for communication.

The main or supervisory module of a task is usually a TSM and is tested by setting up a skeleton of the task to perform synchronization actions. The objective was to verify all task interfaces were developed correctly and task communication behaved as desired.

Once all the TCMs and the TSMs had been tested the next step was to test the individual tasks and gradually integrate them together. Again, the incremental development approach was used to determine the order in which the tasks were developed and integrated.

This process is illustrated by referring to Figure 18. The TCMs and TSMs were developed first. The TCMs consisted of MCMs for the voice command inputs, object packet inputs, displayable object input, and poll request/position orientation, and IHMs for the viewer position, object tracks, object status, and system status. There was one TSM, the recorder controller, which dealt with asynchronous inputs' events from the Voice Command Processor.

Using the incremental development approach and the event sequence diagrams, the critical paths were selected. The first path, the object path, involves receiving, processing, and displaying a stream of object data packets from the network. The second path was the voice command path that received and processed voice commands from the user. To support the object path, functions from the Network Manager, the Object Controller, the Position Sensor Controller, Viewer Position Input Handler, and the Display Manager were all required. The voice path required functions from the Voice Command Monitor, the Voice Command Processor, the Sound Controller, and the Recorder Controller.

In the initial incremental versions for the object path, stubs were developed to simulate receiving and processing object data packets over the network. The voice path contained stubs to receive user voice inputs and control the recorder operations. After unit testing each task in a path, tasks were gradually integrated to form an incremental version of the system with a testable path through the system.

3.4 System Software

The first step in the software development was to evaluate existing software for use with the Synthetic Battlebridge System. Several programs and class structures were adapted for the Synthetic Battlebridge System. Gerken (Gerken, 1991) created a class to track the Polhemus 3SPACE ISOTRAK™ sensor. Simpson (Simpson, 1991) developed software to read and send data across an RS-232 port. Bunderman (Bunderman, 1991) developed a class structure to read in and display models in a polygonal geometry (GEOM) format. Tisdale (Tisdale, 1992) designed a Recorder class to control the Lyon Lamb MiniVas video controller. The class methods he designed emulate the same recorder commands as those sent by the MiniVas' remote controller. Wright and Soltz (Wright and Soltz, 1992) designed a sound generation facility for Macintosh computers.

The Polhemus class was modified to provide user selectable stations and orientation and quaternion data. The initial Polhemus class returned data from all active stations and left it up to the using program to filter unwanted stations. Current structure allows a calling program to select exactly which station it wants data from. There is no real timing advantage in obtaining the data this way since each poll of the Polhemus returns data for all active stations. This was just a personal preference and seemed more object-oriented for the filtering to occur within the class structure.

Most of the code necessary to return orientation data was in place and only the setup calls to the Polhemus and the data structures needed to be added. The primary motivation for adding orientation data was the reduction in transmission time between the Polhemus ISOTRAK™ and the host computer. This reduction is due to the smaller size of the ISOTRAK™ data record, 48 versus 90 bytes. In addition, orientation data translates fairly straightforward into parameters for the Silicon Graphics GL *LOOKAT* routine.

The RS-232 port driver software was used with only a very slight modification to output the port id of the activated port.

The Geometry class underwent a series of transformations. The first modification was incorporating two additional methods developed by Captain Pond. The two methods read in any size geometric model and resize it to a defined unit length and with the model's center at the origin. The second change involved modifying the model loading and rendering methods to provide texture mapping capabilities developed by Captain McCarty. The final modification was also to the model loading and rendering methods and provided transparency levels by alpha specifications in the geometry format.

Both the Recorder and Sound generation software were used without modification. The only alteration required was to add the new sounds for the Synthetic Battlebridge System to the sound class header definitions.

A Glove class was developed by consolidating several C programs from SimGraphics and work done from earlier research into a C++ class structure. Although this software is not part of the current version of the Synthetic Battlebridge System, the class methods can be incorporated into any program with little difficulty.

The next step was to develop the remaining object classes identified during the design process. These were the Voice, Display, Network Manager, and Object Manager classes.

The Voice software was developed for and written on a Macintosh computer. The program is essentially a windows menu manager. The Articulate Systems' Voice Navigator™ system is designed to translate verbal commands into menu commands. In essence, the Voice Navigator™ takes a word it recognizes from the specified dictionary and matches it to a menu item. It then activates the menu item, which appears to the using program as if a mouse had "clicked" on the menu item. Processing of the activated menu item is left to the using program, which in this case is to simply output the activated command to the Macintosh's modem port. The Voice class software on the host Silicon Graphics computer only contains two methods. One method reads the voice commands from the port and the other processes those commands.

The Display class includes methods for initializing and controlling the various display devices, rendering the actual geometric models, and displaying textual information to the user. The initializing method opens the viewing window, sets the viewing perspective, defines the lighting model, and activates the texture mapping capabilities of the Silicon Graphics IRIS computer. Static and dynamic models are rendered based on the 4×4 rotation matrix received from the Network Manager. Displayed textural information includes a context sensitive help menu, viewer location within the environment, system status and event notification, and current information on a selected object.

The Object Manager class provides methods to select an object or model, determine if an object is in the user's field of view, and update object trails and tracks. Objects are identified as selected if they are located within a set radius of the line extending from the user's to a point the user is looking at. In Silicon Graphics terminology this is the view point (VP) and the reference point (RP). Only the first object encountered on this line is selected. To increase rendering performance objects that are outside the user's field of view are culled away. The most computationally efficient method of accomplishing this was to divide the environment into quadrants based on the viewer's location (Figure 20). On the basis of the division in Figure 20, all objects in quadrant C would be culled from the scene. Aircraft trails and missile tracks are maintained for subsequent user directed display.

The Network Manager class defines the actual object record structure and the methods to transmit and receive object packets over a network. This software was developed by Captain Sheasby (Sheasby, 1992) and is transparent to the Synthetic Battlebridge System. In cooperation with Captain Sheasby, I developed several additional methods for use exclusively with the Synthetic Battlebridge System. These methods allow the direct manipulation of flags within the object record structure to control the display features of the individual objects.

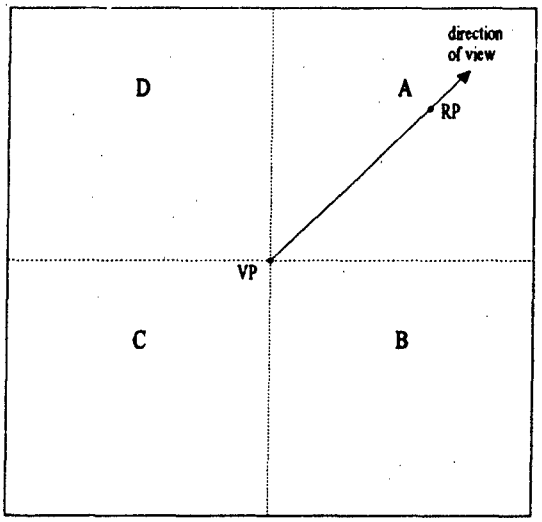

Figure 20. Viewing Quadrants

The main program or driver for the Synthetic Battlebridge System instantiates the various class objects, initializes the system variables and class objects, forks the tracking and network processes to a separate processor, monitors the keyboard and voice ports for input, provides aural and visual feedback, and obtains and processes the objects for rendering.

The overall structure of the driver program has remained constant despite several hardware modifications and feature enhancements. I attribute this to the extensive design work using the DARTS methodology and the incremental prototype that gradually evolved as additional classes and features were incorporated.

3.5 System Environment

Most real-time applications are developed for microcomputer, multiprocessor computer systems, or embedded systems whose environments are typically poor in software development tools. In most cases, real-time systems are developed on host

platforms and then cross-compiled or cross-assembled to the target platform. Testing is then done on the target platform.

The approach taken on the Synthetic Battlebridge System and several related projects was to do as much testing as possible on the host platform where more development tools were available. We also tried to match the host and target platforms as much as possible. For example, we ensured both platforms had the same compilers, operating systems, and external hardware. This approach has worked extremely well for testing since software is thoroughly tested on the host before being tested on the target machine.

My experience with the Synthetic Battlebridge System is not typical of most real-time systems. Most often the operating system requirements for the host are completely different from those of the target system.

The key elements in the testing of real-time systems are task integration and timing. It is essential that the operating system support the inter-task communication and task synchronization primitives. This is usually done by emulating the target platform's performance characteristics on the host platform.

Figure 21 displays the Synthetic Battlebridge System's current hardware configuration using the Polhemus LookingGlass™ head-mounted display. The host computer is a Silicon Graphics IRIS 4D/440VGXT workstation, with four processors and 64 MB of main memory. The computer is responsible for executing the software to create and maintain the synthetic environment, driving the display devices, and communicating with all peripheral devices.

Peripheral devices connect to the host computer and pass data between the device and the host through RS-232 serial ports. These devices display the environment, position the user in the environment, and allow communication and feedback with the host.

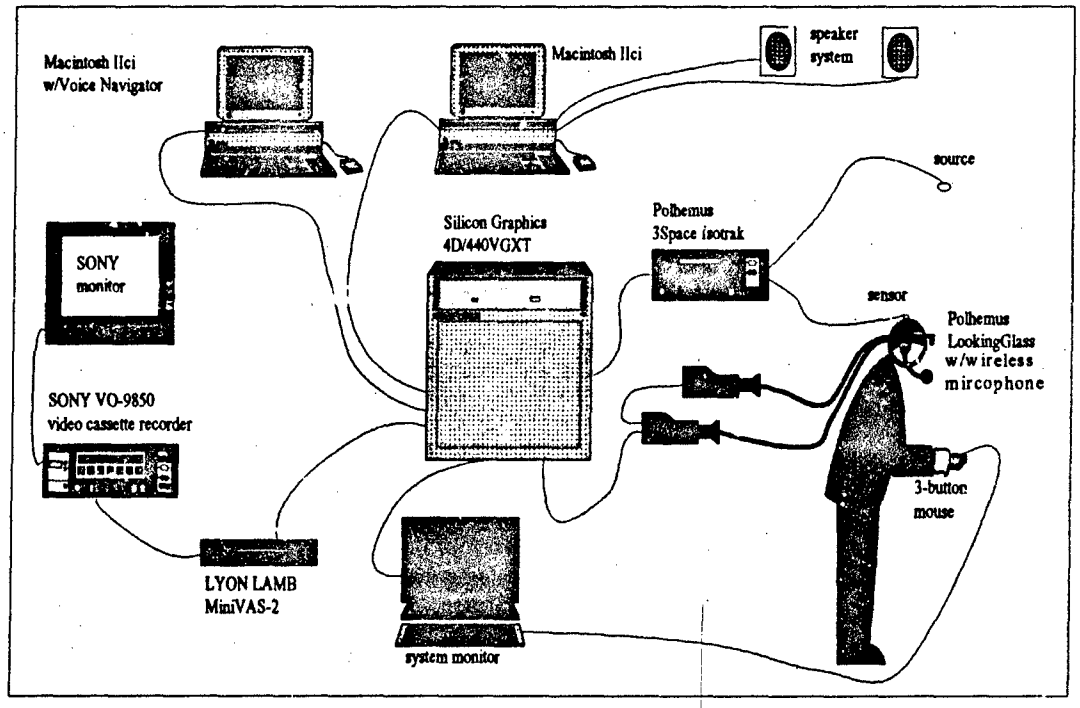

Figure 21. Hardware Configuration w/Polhemus LookingGlass™

A single NTSC video signal is fed to a Sanyo Liquid Crystal Color Video Projector. This signal is echoed to a second projector and projected onto the end of a fiber optic bundle. The opposite ends of the fiber bundles are tapered and fused onto a 25 ×20mm rectangular glass that reflects an image onto a pair of half-silvered mirror glasses.

Navigation within the environment is provided with a 3-button mouse and position tracker. Mouse buttons provide forward and reverse movement and restoration of initial positioning.

To allow a user to record their interactive sessions, the Synthetic Battlebridge System uses a Sony VO-9850 3/4-inch NTSC video cassette recorder. Since the Silicon Graphics system monitor cannot display NTSC video, a Sony television monitor allows non-participants to view the synthetic environment while a user views the environment in the LookingGlass display. Viewers can watch the current session or previous sessions recorded on tape. The monitor connects to a second output port on the recorder.

To provide remote control of the video cassette recorder, the recorder is connected to the IRIS computer by way of a Lyon Lamb MiniVas-2 video tape recorder controller. The MiniVas allows for either frame-by-frame or continuous recording of video images.

Users must have a means to input commands into the system when they are immersed in the synthetic environment. They must also be able to control some of the viewing conditions in the environment. Two different methods of providing user input to the Synthetic Battlebridge System were evaluated: voice commands, using the Articulate Systems' Voice Navigator™ connected to a Macintosh IIci computer, and hand gestures using a VPL DataGlove™ Model 2. The Voice Navigator™ is a relatively inexpensive method of adding voice capabilities to a synthetic environment. It connects to the modem port on a Macintosh computer and Voice Navigator™ software allows menu items to be captured and activated by voice. The Synthetic Battlebridge System then translates these captured menu commands into voice commands and sends them to the host computer over a 9600 baud RS-232 line. Both the Voice Navigator™ and the VPL DataGlove™ require extensive user training. Users must train the Voice Navigator™ to recognize the way they pronounce certain words, while the VPL DataGlove™ requires users to train the DataGlove™ system to recognize gestures. However, the Voice Navigator™ seems much more reliable in accepting verbal commands than the DataGlove™ is in accepting gestures. The Voice Navigator™ also allows the development of a more intuitive command set. The Voice Navigator™ has no theoretical limit to the number of words that can be stored in a user's dictionary, however system memory and recognition performance limit actual user vocabularies. Current user dictionaries for the Synthetic Battlebridge System are approximately 30 verbal commands. This is a significant improvement over the DataGlove™ which can only recognize a maximum of 10 gestures. My experience on the Synthetic Battlebridge System and earlier projects indicated the DataGlove™ worked extremely well with only two or three extremely exaggerated gestures. For these reasons,

I decided to refrain from using the VPL DataGlove™ and use the Voice Navigator™ as a sole method of inputting user commands.

For additional flexibility any of the voice commands can be input from the IRIS keyboard. A complete listing of voice and keyboard commands can be found in the user's manual in Appendix B.

A second Macintosh IIci computer is connected over another 9600 baud line to a RS-232 port to provide audio feedback and significant event occurrence notification. The audio output port of the Macintosh can drive either a set of stereo headphones or a pair of stereo speakers.

A Polhemus ISOTRAK™ system is used to determine the position and orientation of the user. The system contains a single sensor source pair and returns the sensor's position and orientation in reference to the Polhemus' stationary source. Position values are expressed as x, y, z coordinates, while orientation values are the euler angles for azimuth, elevation, and roll. The ISOTRAK™ provides extremely accurate position data within an operational hemisphere of about 40 inches. This accuracy declines rapidly from 40 to 60 inches and values beyond this range are unreliable.

Figure 22 shows an alternative hardware configuration using the Fake Space Labs BOOM2M™ system. This configuration functions the same way as the LookingGlass™ option with a few exceptions.

Since the BOOM2M™ uses the standard video produced by the IRIS workstation, no NTSC video signal is generated so the Sony video recorder can not be used. The BOOM2M™ also provides all necessary tracking data through its mechanical linkage therefore, the Polhemus ISOTRAK™ tracker is not used. The BOOM2M's buttons replicate the movement features provided by the mouse.

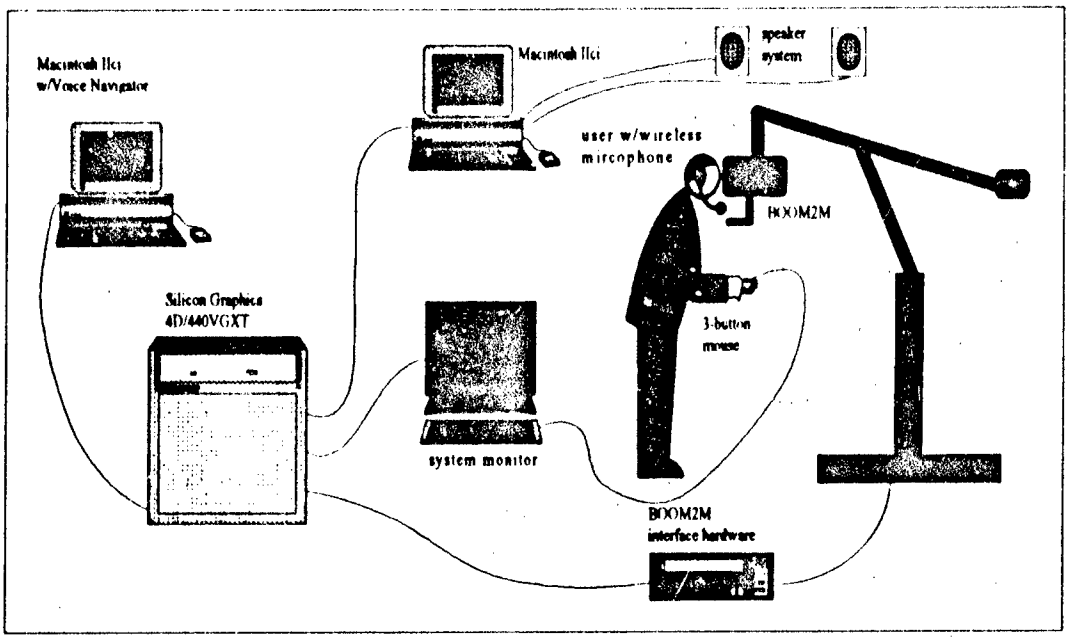

Figure 22. Hardware Configuration w/BOOM2M™

3.6 System Operation

During system initialization, objects are instantiated for the Display, Object Manager, Network Manager, Polhemus, Recorder, Sound, and Voice Command classes. An instance of each available geometric model is created and the associated file is read into the system. Settings for the default and nine user defined views are defined. Following this, the Polhemus object opens an RS-232 port to the Polhemus ISOTRAK™ and prepares the device for data transmission and the Recorder object opens another RS-232 port to the Lyon Lamb MiniVas controller. Both of these devices require establishment of a two way communication link.

Following successful initialization of the ISOTRAK™ and the MiniVas, the Sound and Voice objects open RS-232 ports to the two Macintosh computers. The Network Manager is then initialized and forked to a separate processor. If the BOOM2M™ is used in place of the Polhemus ISOTRAK™, a process to continuously read the position and

orientation data is forked to a separate processor. Following the initialization phase the main loop of the Synthetic Battlebridge System is entered.

The main operating loop of the driver program is straightforward:

```
while not done {
        determine viewer location and direction of view
        render static objects
        display requested textual data
        get, process, and render dynamic objects
        get and process user commands
}
```

Viewer location and direction of view are based on the position and orientation data obtained from the tracking device. Offsets are used to accommodate any additional viewer movement indicated by keyboard, mouse, or BOOM™ buttons. If the viewer is attached to an object the position data from the tracking device is ignored and the position of the object is used to determine user location. Orientation data from the tracking device is still used to determine the user's orientation.

Static objects displayed by the Synthetic Battlebridge System include terrain and wall charts. The terrain models are constructed from Defense Mapping Agency (DMA) Digital Terrain Elevation Data (DTED) (Figure 23). The current terrain model covers an area of 128,000 by 288,000 meters and is divided into six areas (Figure 24). Multiple levels of detail are based on elevation marker spacing. Models are available with spacing of 4,000, 8,000, and 16,000 meters. Section V has an additional level with 2,000 meter spacing. Displayed detail level is based on the distance from the viewer to the center point of the terrain section.

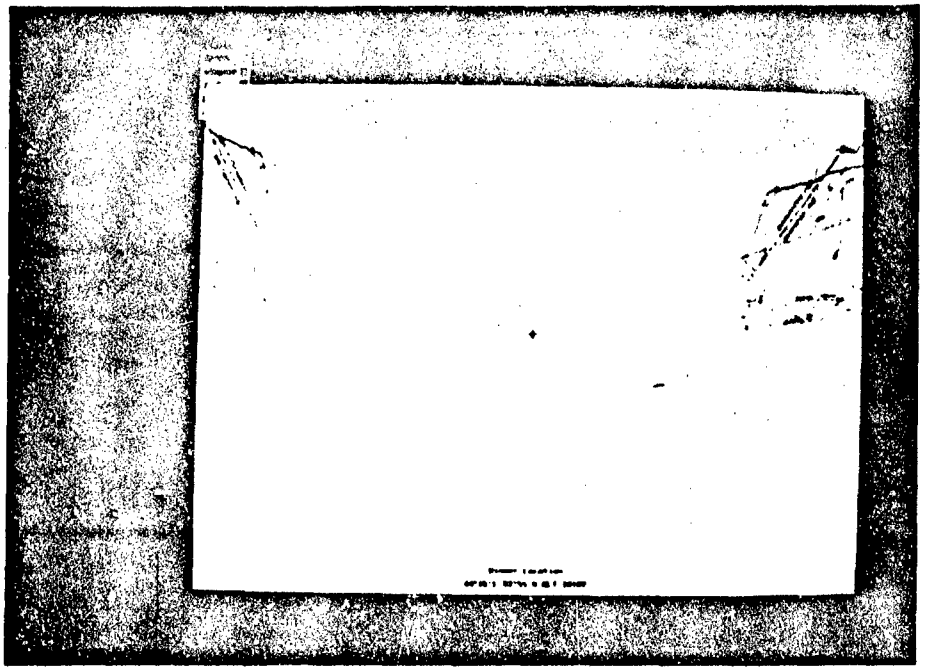

Figure 23. Terrain Display

0, 128000 IV	V	288000, 128000 VI
I 0, 0	II	III 288000, 0

Figure 24. Terrain Sections

Texture mapped images of scanned DMA charts are displayed as wall charts on the north and south sides of the terrain (Figure 25). The wall charts are mirror images of each other and a marker is located on each to indicate the position of the viewer on the terrain.

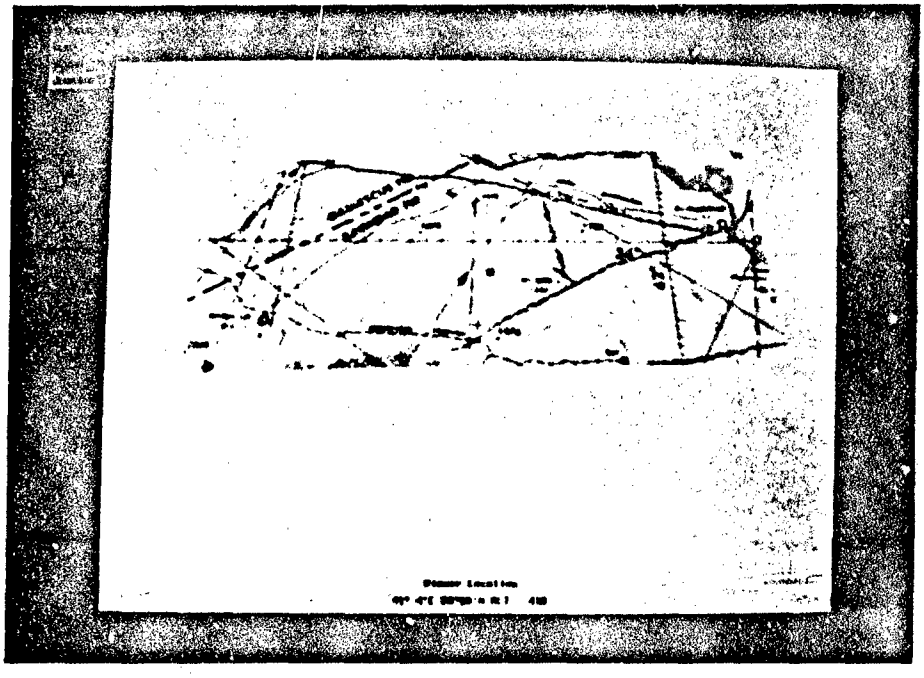

Figure 25. Texture Mapped Wall Chart

Textual information is displayed based on the status of system flags set by user commands. Information includes the context sensitive help menu (Figure 26), object or vehicle information (Figure 27), and status icons (Figure 28).

The help menu displays only context sensitive commands and dynamically updates based on additional commands or events. The menu will remain displayed until the user turns it off. The vehicle information display shows current data on the selected vehicle. This data is derived from the network object record and contains network id, vehicle type, country, location, and altitude. System status icons show the user which display or system options are active.

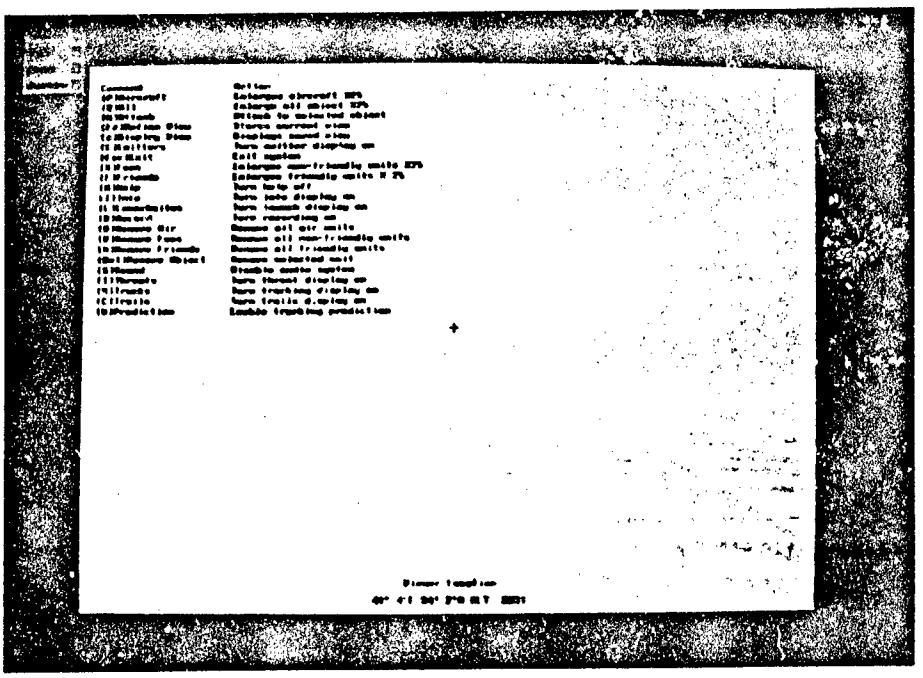

Figure 26. Help Menu

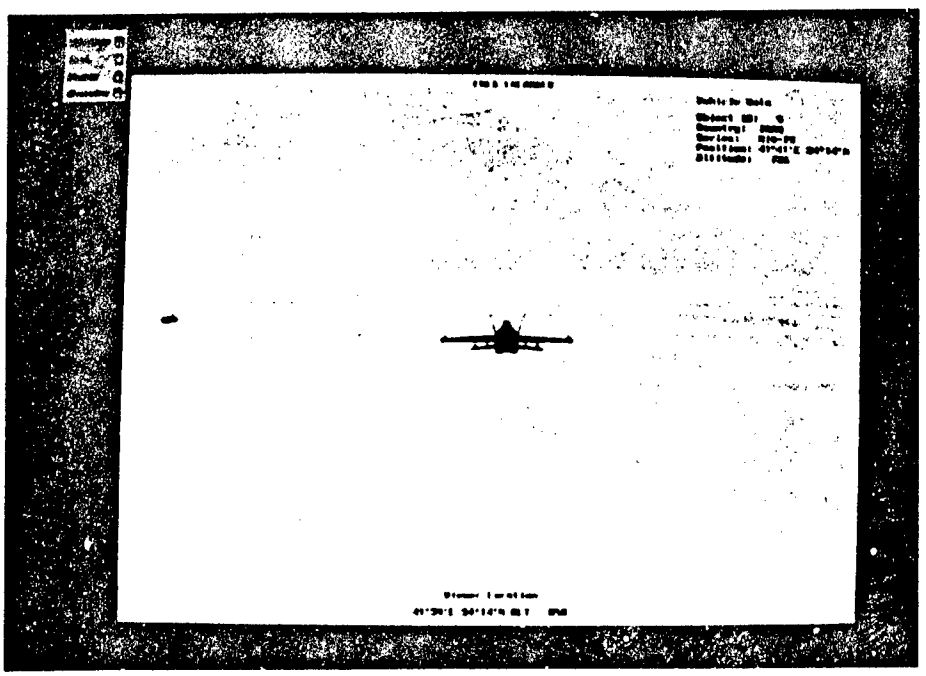

Figure 27. Vehicle Data

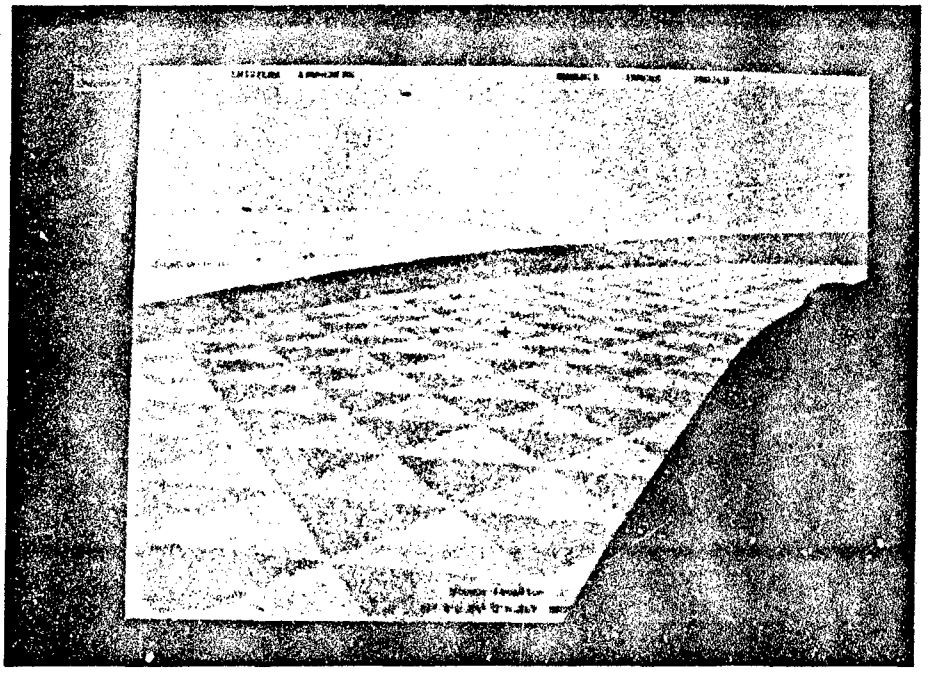

Figure 28. Status Icons

A list of active objects or vehicles is obtained from the network manager and processed based on user inputs. Objects are process sequentially, with updating the position data for aircraft trails and missile tracks as the first step. The next step is to cull away objects not in the user's field of view. The final processing step is to assign a geometric model to the object based on object type and its distance from the viewer. Each object is then rendered based on system status flags previously set by the user's commands.

During the rendering portion of this processing additional display option flags are evaluated and displayed if active. Display options include threat envelopes (Figure 29), radar envelopes (Figure 30), aircraft trails (Figure 31), missile tracks (Figure 32), and missile launch locations (Figure 33).

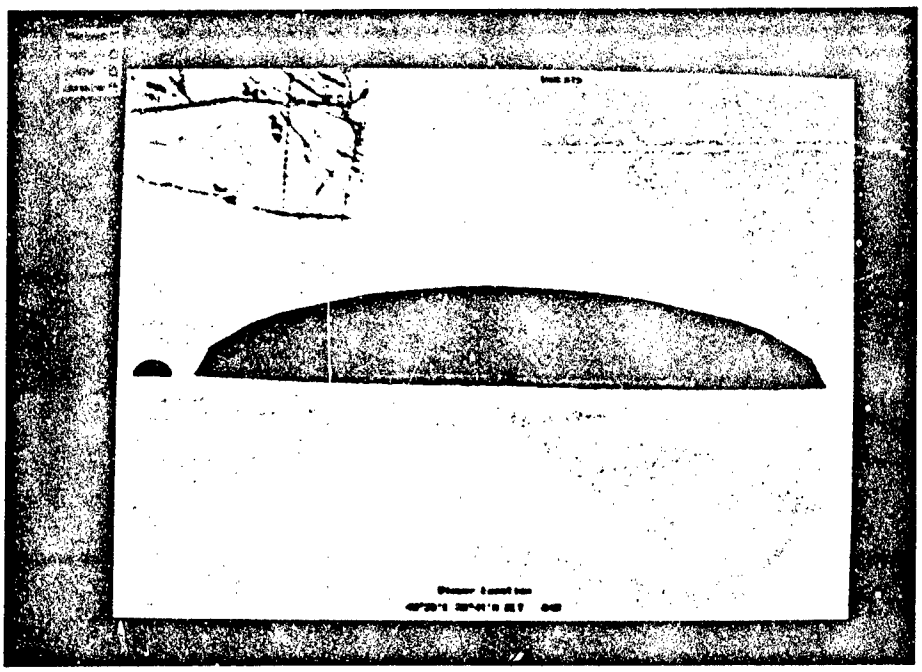

Figure 29. Threat Envelope

Threat envelopes are displayed for active surface-to-air missile systems (SAMS) and anti-aircraft-artillery (AAA) vehicles. Envelopes are derived from unclassified, published data and assume maximum capabilities without consideration of terrain or atmospherics.

Radar envelopes are displayed for active, emanating SAMS, AAA, and radar systems. Envelope criteria is the same as for threat envelopes.

Aircraft trails are displayed for all active aircraft and missiles. Trails show the flight path of the vehicle over the previous fifteen seconds. This length is not a user selectable option however, the code could be modified to handle any time span up to the size of the array (currently 50 position reports). Missile tracks are displayed for all active and deactive missiles. Tracks show the entire trajectory of the missile from initial activation to deactivation or impact. The tracks remain viewable for the entire user session.

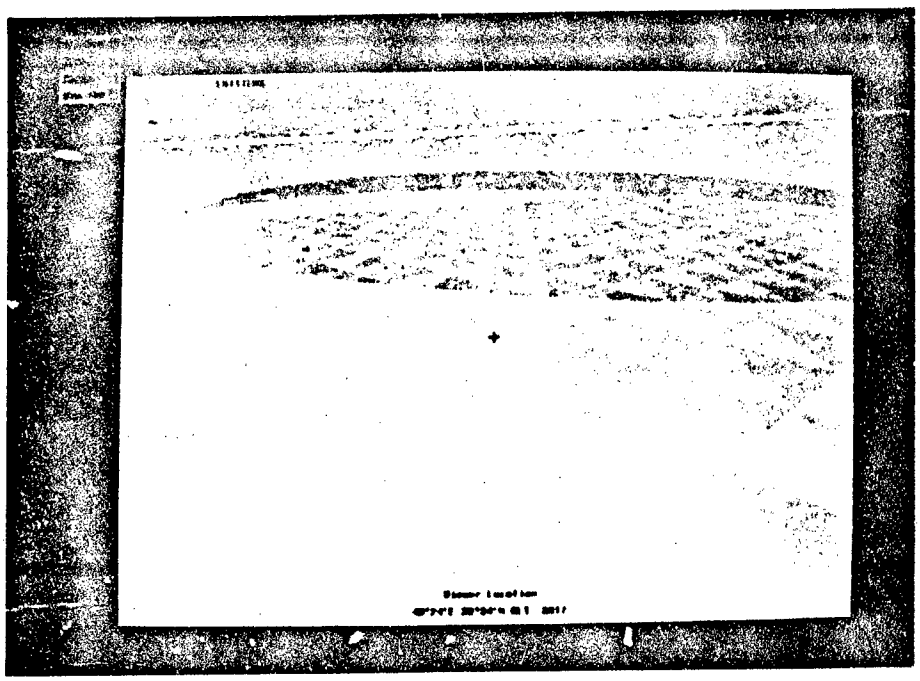

Figure 30. Radar Envelope

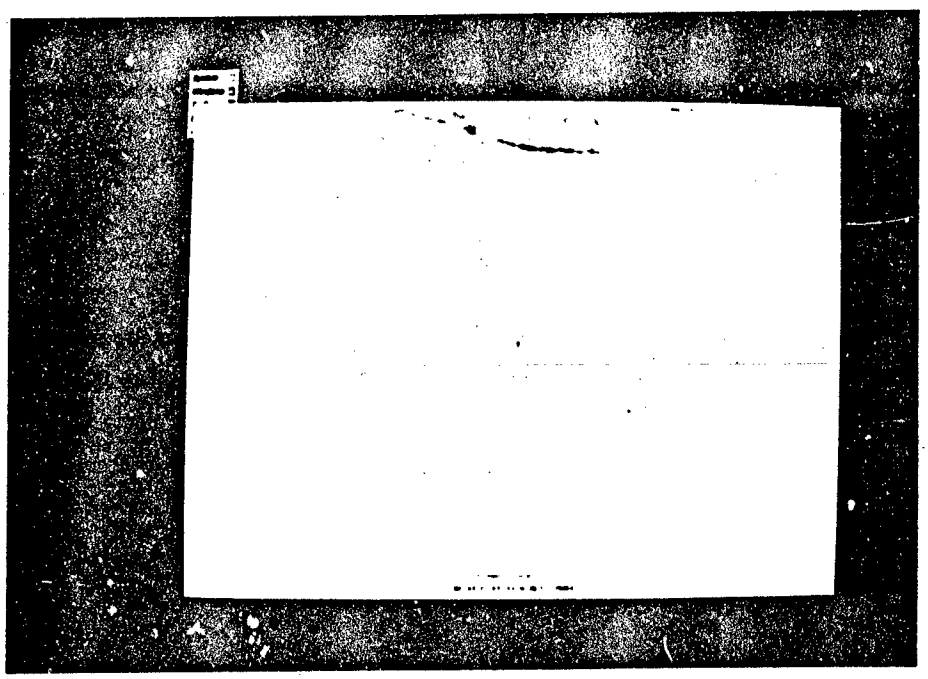

Figure 31. Aircraft Trails

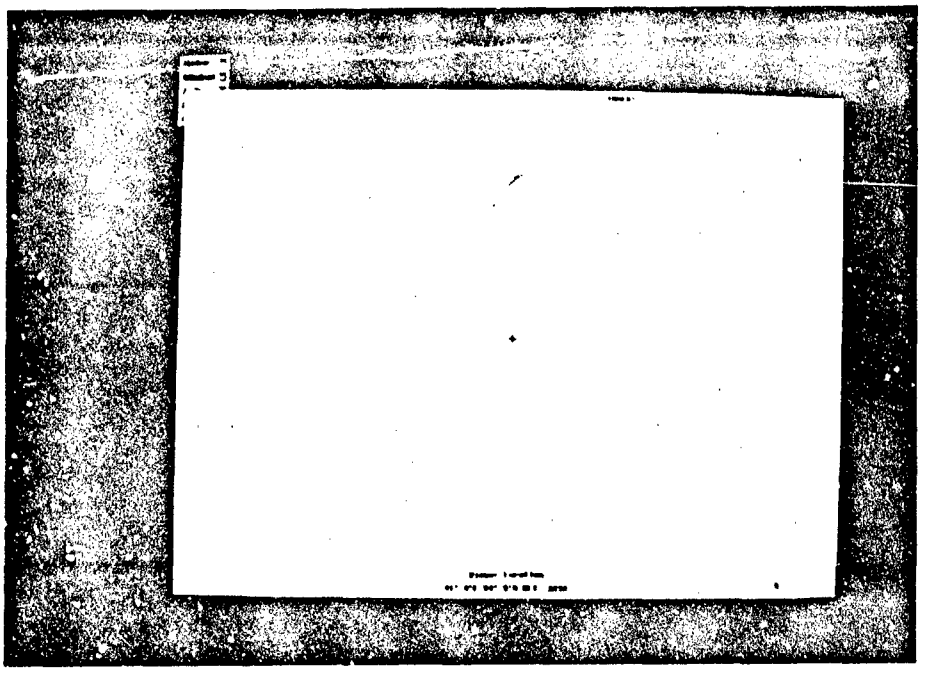

Figure 32. Missile Tracks

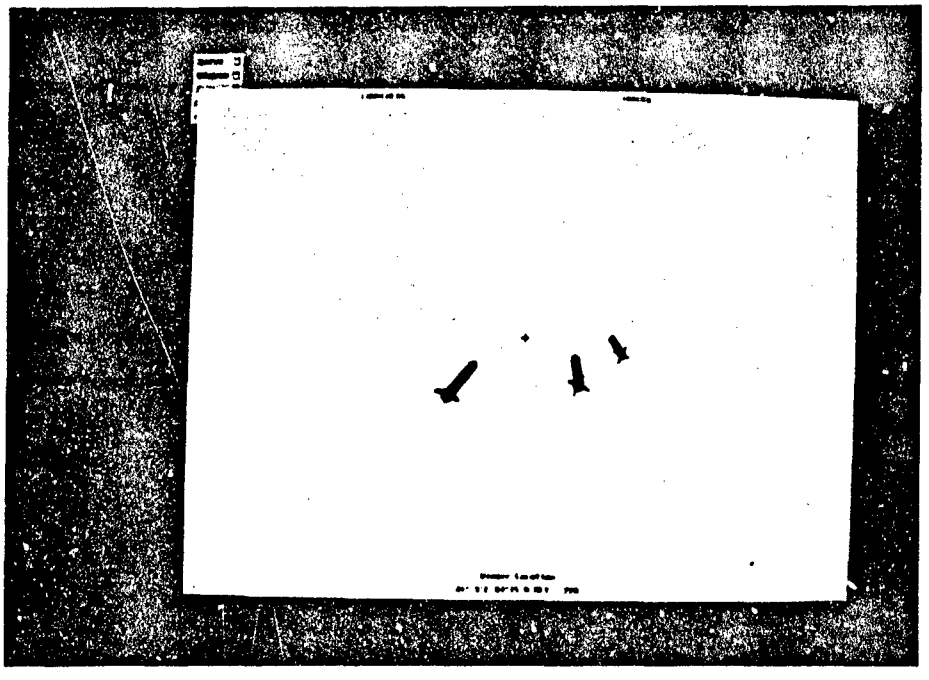

Figure 33. Missile Launch Locations

Missile launch locations are displayed for all active and deactive missiles. The locations are derived from the initial position of the missile with an icon corresponding to the missile type placed at that position.

User commands input either verbally or by keyboard are used to set system status flags. Confirmation feedback of each command is provided by a displayed textual message and aurally through the sound generation facility on a Macintosh. A complete listing of system commands is found in the user's manual (Appendix B).

3.7 Conclusion

The DARTS design approach extends the Structured Analysis/Structured Design methodology to address the requirements of real-time systems. This method leads to a highly structured, modular system with well-defined interfaces and reduced coupling between tasks. The application of the DARTS design method to the Synthetic Battlebridge System has proven very successful. Although requirements specification changes involving both hardware and software have occurred, no major changes to the system structure or task interfaces have been required.

Prototyping was used in the Synthetic Battlebridge System to generate an evolutionary prototype that was the result of incremental development. Since this prototype gradually evolved, it was necessary to follow the software life cycle, with requirements specification, system design, and detailed design phases preceding the coding and testing.

I found the incremental development approach resulted in the availability of an early, morale-boosting, version of the system. It also had the effect of spreading out system integration testing over a wider time span.

IV. Results and Recommendations

4.1 Introduction

The goal of this research was to develop a synthetic environment to display objects dispersed over a large area. The intent was to build a prototype that could be used to explore various interactive devices and techniques. It was not the purpose of this research to develop a fully capable commercially acceptable product. This chapter presents observations of work performed, problems experienced during the development, and recommendations for further research and development.

4.2 Observations

The Synthetic Battlebridge System allows users to navigate through the battlefield and provides limited interaction with objects or vehicles dispersed over the battlefield. Users can view objects at increasing levels of detail and observe the battlefield from the object's perspective.

Using the DARTS design methodology coupled with object-oriented programming and the concept of rapid prototyping proved to be an effective method of defining system performance and requirements. The DARTS method showed which processes should or should not be grouped together and which processes were the most likely candidates for executing in parallel. The use of object-oriented programming also proved to be extremely helpful. Once a portion of the implementation was designed and programmed, I no longer had to worry about that portion. I was able to break the overall system design into smaller, less complex concepts that were substantially easier to design and program. For example, after initial modification of the classes for the Geometry, Polhemus, Recorder, and Sound objects, I never had to change them.

The Voice Navigator™ system also proved very successful. Most users felt that the voice interaction provided a much more intuitive means of inputting user commands than other evaluated methods. Once the initial learning curve in using the Voice

Navigator™ had been overcome, software development with the system was elementary and very flexible. This flexibility proved extremely valuable in that voice interfaces were added to both the Satellite Modeler (Pond, 1992) and the Texas Instruments Omniview Interface (Hobbs, 1992) with only very minor modifications.

4.3 Problems Experienced

As with any research project of this size, numerous problems were experienced throughout the design and development phases. The problems encountered are presented in the order I consider the most consequential.

The one consistent obstacle throughout this research, and the one that seemed to give me the most difficulty, was the lack of proper documentation on most of the hardware. This included everything from no documentation to misleading or incorrect data concerning the performance and capabilities of the equipment itself. Most problems were overcome by talking directly to company engineers, fellow students, and sheer perseverance.

The slow frame rate is the most noticeable problem with the Synthetic Battlebridge System. Current frame rates fluctuate between 10 to 12 frames per second. Most attempts to increase this rate have had limited success. The most influential factors on the poor frame rate have been the forking of the network manager to a separate processor, the amount or number of times text is written to the display and the number of polygons rendered. By executing the network manager on a separate process the effective frame rate was doubled. Initial execution of the network manager was done sequentially during the main operating loop of the driver program. When a call was made to get the current objects in the environment, the network manager would then read and process the vehicle data packets stored in the buffer since the last request. Current software now continually processes the vehicle data packets and when requested provides a list of active vehicles.

The textual information written to the display decreases the frame rate by approximately 20%. However, most users feel the data displayed is critical and worth the performance tradeoff. One possible alternative may be to make the user location data a command toggle.

The number of polygons displayed is dependent on the number of objects in the scene. Culling those objects not in the user's field of view did improve performance but, there can be a very noticeable difference in performance depending on the scene. Some users find the varying frame rate annoying and would prefer a consistent rate even if it is slower.

The most dramatic effect on frame rate performance, other than executing the network manager in parallel, came from using multiple levels of detail for the objects displayed. Exact distances to change from one level to the next depend on the user's intentions. For this reason the changeover distances were made global variables and placed in the system header file. Unfortunately, most of the models available do not have muitiple levels of detail.

Along with this lack of detail levels, there is a general lack of models. The network manager recognizes over one hundred different vehicle types however, models are only available for less than thirty vehicles with virtually no models for "red" forces. Many of the models available are, at best, of questionable use. The scaling and dimensions of some of the models are incorrect and the colors are not indicative of actual appearances. In most instances, equivalent "blue" models are used to represent "red" forces. For example, an F-15 model is rendered for a MIG-25. Another difficulty encountered with the existing models was the different orientations used.

The world coordinate system used by the Synthetic Battlebridge System is a right handed coordinate system with positive x to the viewer's right, positive y up, and positive z coming out of the screen toward the viewer (Figure 34). Most models had to be rotated in one or two axes to match this orientation.

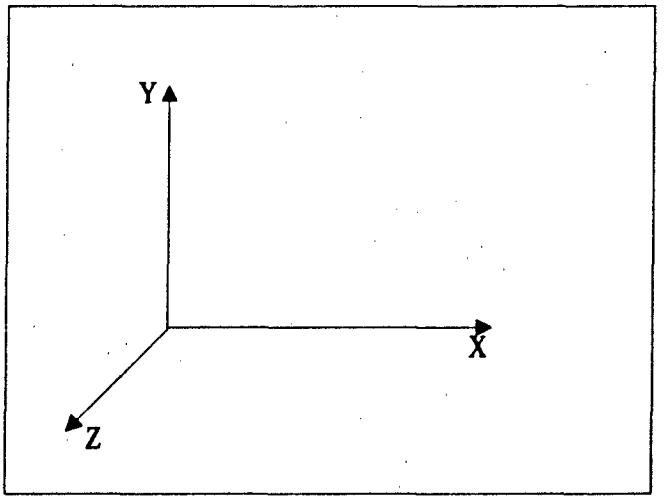

Figure 34. World Coordinate System

This orientation problem is also visible in the rotation matrix received from the SIMNET data packets. In SIMNET, the positive directions of the coordinate axes are: x right, y in or away from the viewer, and z up. The 3×3 SIMNET rotation matrix is shown in Figure 35.

CH * CR + SH * SP * SR	-SH * CR + CH * SP * SR	-CP * SR
SH * CP	CH * CP	SP
CH * SR - SH * SP * CR	-SH * SR - CH * SP * CR	CP * CR

where:

SH = sine of heading	CH = cosine of heading
SP = sine of pitch	CP = cosine of pitch
SR = sine of roll	CR = cosine of roll.

Figure 35. SIMNET 3×3 Rotation Matrix

Depending on the heading and roll of an object, the appearance may or may not be correct. For example, if the object is heading due north (azimuth zero) and the roll is zero

the displayed orientation of the object is correct. As the azimuth and roll change the object's orientation becomes erratic.

Another problem regarding the SIMNET data is the lack of pertinent information for many vehicles. The SIMNET data set was primary designed for ground vehicles and has been used exclusively for distributed simulations. The Synthetic Battlebridge System is not limited by simulated data. The system has no knowledge of the data source, therefore the data can come from multiple sources, real or simulated. An example of this conflict is the radiation data packet. In SIMNET this packet is sent by the emitting unit and includes a list of the objects or vehicles the unit sees with its radar. Real intelligence data could only indicate whether a radar is active and not what objects or vehicles they see. There are some exceptions to this depending on the source of the data. Another example is the lack of unit affiliation and tasking information. If this information were available, the Synthetic Battlebridge System could be enhanced to provide mission data such as flight path, estimated fuel reserve, target, time over target, and weapons delivery.

Additional problems surfaced when modifying the geometry class structure to include transparency. To render transparent radar and threat envelopes, alpha channel capabilities were added to the renderer. Although this provided excellent results for the envelopes it had the side effect of causing other objects, or parts of other objects, to become transparent.

Without using the alpha channel, the envelopes were rendered using *screen-door transparency* with meshs of either 1/4, 1/2, or 3/4 of the density. I found this unacceptable because of the shadow casting effect on the back side of the envelopes when the z-buffering option was active. In addition, there was a positive to having the terrain transparent; objects positioned under the terrain, either by misinterpreted position data or elevation marker spacing, could still be seen.

The terrain itself also presented an obstacle. The terrain data used was not compatible with any programs or translation systems available. The current terrain models

used by the Synthetic Battlebridge System were developed manually from DTED files. This was an acceptable alternative only because the limited coverage area of the models.

The remaining problems were all hardware related The command recognition of the Voice Navigator™ was initially only about 85% accurate. I was able to improve this by reducing the confidence recognition level down from the recommended 90% to 80% and redefining the command vocabulary to ensure as much word distinction as possible. This modification improved the recognition accuracy to slightly over 90%. By adding an omni-directional or noise canceling microphone, command recognition accuracy increased to over 97%.

Two problems arose with the Polhemus 3SPACE tracking systems. First, I was unable to execute the ISOTRAK™ on a separate processor for an extended period. Within two to three minutes of initialization the ISOTRAK™ would freeze and cease responding to poll requests. By reducing the number of poll requests to less than ten a second the ISOTRAK™ would function normally however, the advantage of executing the system in parallel was lost.

The second problem occurred with the newer Polhemus FASTRAK™ system. Although the Polhemus class methods worked with the test program, I was unable to integrate the FASTRAK™ into the Synthetic Battlebridge System.

The Polhemus LookingGlass™ head-mounted display has not proved as functional as originally expected. The weight of the fiber bundles requires additional support and the wearer must remain seated to use the system. The alignments and focusing controls are awkward and difficult to adjust and the 42° field of view is too narrow to give the user a sense of immersion in the environment. Another more serious concern with the LookingGlass™ is the severe eye strain and headaches reported by several users after as little as five minutes of use.

4.4 Further Research and Development

Several improvements can be made to enhance the Synthetic Battlebridge System. The most significant impart of system capability could be gained by continuing to implement the SIMNET data set. Many of the vehicles and data packets available in SIMNET are not recognized by the network manager. Continual expansion of the SIMNET and DIS data sets would allow additional features to be implemented.

To increase the usefulness of the Synthetic Battlebridge System the terrain database must be expanded. Mackey at the Naval Postgraduate School has developed and implemented a terrain paging algorithm (Mackey, 1991). The terrain paging expands the terrain area available and generates a multi-resolution dataset.

The model library used by the Synthetic Battlebridge System needs to be expanded or alternative rendering methods could be implemented that support formats other than the AFIT GEOM.

Addition of stereoscopic displays and spatially localized sound would provide additional depth cueing for a user.

Replacement of the Polhemus ISOTRAK™ with the newer FASTRAK™ would provide additional range and accuracy for user position and orientation.

4.5 Conclusion

This chapter presented the results of the research and development done on the Synthetic Battlebridge System. Problems experienced and how they were resolved or worked around were discussed. The last section gave my recommendations for future research and enhancements to the Synthetic Battlebridge System.

The overall results of this study indicate that although the Synthetic Battlebridge System requires a significant amount of work, it is a useful tool for displaying and interacting with objects dispersed over a large volume of terrain. The system can provide a commander with a better understanding of the spatial orientation and distribution of

elements on the battlefield. And with this understanding comes an increase in the commander's situational awareness, a crucial part of the critical decision making process.

Appendix A: Software Class Structures and Methods

A. Display Class

```
struct geometry {
        GeomClass    *gname;      // geom file
        Matrix       position;    // object position
};
```

Display_Class();

Constructor for display class.

void display_clear();

Clears the display. Used for double buffering.

void display_init();

Initializes all display hardware and associated viewing parameters.

void display_emitters(object*, geometry*);

Displays radar envelopes of all active radiating vehicles.

void display_help();

Displays context sensitive help menu in upper left corner of display.

void display_info(object*);

Displays current system data of selected vehicle.

void display_object(geometry*);

Displays static objects in environment. Used for terrain and wall charts.

void display_launchers(geometry*, track*),

Displays missile icon at position missile was initially detected.

void display_status_icons(Point);

Displays active status toggles and textual feedback of user voice commands.

void display_selector();

Displays selector symbol (+) in center of display.

void display_threats(object*, geometry*);

Displays threat range envelopes of all active SAM and AAA vehicles.

void display_trails(trail* object*);

Displays aircraft trails or flight path for last 10 seconds.

void display_tracks(track*);

Displays missile track or flight path from initial detection till impact or deactivation.

void display_viewer_position(Point);

Displays viewer position and view selection in the bottom center of the display.

void display_world_object(object*, geometry*, int, float);

Displays all active vehicles.

B. VehicleObjectManager Class

```
struct object {
        int                 object_id;      // network id
        entityCountry       country;        // country of object
        entityEnvironment   obj_type;       // type of object
        entitySeries        obj_series;     // series of object
        boolean             active;         // activate toggle
        boolean             display;        // display toggle
        boolean             selected;       // select toggle
        boolean             threat;         // threat toggle
        boolean             radiating;      // radiate toggle
        boolean             friendly;       // friend toggle
        Matrix              position;       // object position
};
```

vehicleObjectManager()

Constructor for the vehicleObjectManager class.

int get_world(object*);

Returns an array of all active vehicles.

void set_display(boolean);

Sets the display flag to the boolean value.

void set_friend_display(boolean);

Sets display flag of all friendly vehicles to boolean value.

void set_foe_display(boolean);

Sets display flags of all non-friendly vehicles to boolean value.

void set_object_display(int, boolean);

Sets display flag of selected vehicle to boolean value.

void set_type_display(entityEnvironment, boolean);

Sets the display flags of all vehicle of environment type to boolean value.

void set_select(int, boolean);

Sets the select flag of identified vehicle to boolean value.

void init_world();

Initialize vehicle array. Used only in non-network mode.

C. Object Class

Object_Class();

Object class constructor.

int select_object(object*, Point, Point, int);

Determines if an object is on or near the line define by the VP and RP points, returns object id.

boolean object_in_view(object*, Point, Point);

Determines if object is in field of view based on quadrant formed by extending the VP and RP points.

void update_trails(object*, trail*);

Updates the last position of the vehicle object. Positions maintained for last 10 locations. Overwrites oldest location if no space is available.

void update_track(object*, track*);

Updates the object track. Maintains position information for entire duration object is active.

D. Voice Class

Command_Class();

Command class constructor.

void open_Command_Port();

Opens RS-232 voice command port at 9600 bauds.

int read_Command();

Reads voice commands from RS-232 port.

~Command_Class();

Command class destructor.

Synthetic Battlebridge System (SBS) User's Manual

1.0 Overview

SBS is a SIMNET 6.6.1 compatible Silicon Graphics IRIS 4D workstation based synthetic environment. SBS is designed to allow limited interaction with vehicles displayed over a large volume of terrain. The objective of this manual is to provide a broad overview of the nature and capabilities of SBS, plus a specific "how to" guide to rapidly orient both the novice and experienced user.

2.0 Capabilities

SBS is limited to 500 active entities or vehicles at any given time. The network manager uses an array structure to maintain this list of vehicles. After a vehicle is deactivated, the slot is available for reuse. If there is not a slot for the incoming vehicle the network manager reports the error and continues to process other vehicles.

The terrain database covers an area of 128,000 × 288,000 meters in northwest Iraq. All object and vehicle models are in the AFIT GEOM format.

3.0 Hardware Requirements

There are several different hardware configurations possible with SBS. The host computer is a Silicon Graphics IRIS 4D/440VGXT with a minimum of 64Mbytes of memory and 1 to 6 available RS-232 ports. Apple Macintoch IIci computers provide voice input using the Articulate Systems' Voice Navigator™ and audio feedback using the Sound Generation Facility software and a pair of external speakers.

Caution: The Voice Navigator System must be installed and activated when the Macintosh is initially turned on. If you attempt to connect or activate the Voice Navigator System after the Macintosh is on serious damage to the Voice Navigator™ could result.

The display may be the system CRT, a head-mounted system like the Polhemus LookingGlass™ with user position and orientation provided by a Polhemus 3SPACE ISOTRAK™ or the Fake Space Labs BOOM2M™ system. A Sony VO-9850 3/4-inch NTSC video cassette recorder connected through a Lyon Lamb MiniVas controller is required to make video recording.

These are the RS-232 port assignments on the Silicon Graphics IRIS 4D:

2 - Voice input from Macintosh	45 - BOOM2M™
3 - Polhemus ISOTRAK™	46 - VPL DataGlove (not used)
4 - Sound output to Macintosh	47 - Recorder control to MiniVas

4.0 Directory Structure

The SBS directory structure is self-contained with the exception of the standard header files located in **/usr/include** and the network management software located in **/usr/eng/ssheasby/CSCE799/v1_00**. The makefile contains only relative path addressing and all models are in the subdirectory **models**. All software has been compiled under Irix 4.0.

5.0 Command Line Arguments

In order to provide the maximum flexibility several command line switches are included. For the most part command switches allow the system to be configured with different display options. The program is started by entering **./sbs [-option(s)]** where

Option	Purpose
-b	Use the Fake Space Labs BOOM2M™ display
-n	NTSC video mode (use with all head-mounted displays)
-r	Use video tape recorder

NOTE: the system can only record when in NTSC mode, so the -r option would only be used when the -n option is specified.

To use voice command inputs the SBS program on the Macintosh must be running. The program is started by double clicking the mouse pointer on the SBS icon.

To use the audio feedback option a second Macintosh must be running the Sound Generation Facility (SGF) software. To execute the SGF program double click the mouse pointer on the SGF icon.

6.0 Run-Time Interaction

Once SBS is running, you can control your position by either using the mouse buttons or the arrow keys on the IRIS keyboard. If the -b option is used the BOOM2M™ buttons are also active. Your initial position is in the center of the terrain or battlefield and this is always the default reset location.

a. Issuing Voice Commands

All voice commands are preceded by the word "COMPUTER" and must be as clear and distinct as possible. In order to achieve a reliable command recognition you must build your own vocabulary file and train the Voice Navigator™ system to recognize your voice. See the Voice Navigator™ Owner's Guide for detailed instructions.

b. Getting Help

A context sensitive help menu is available upon request by giving the verbal command "HELP" or pressing the "H" key. The menu only lists those commands that are available and will dynamically update as you issue new commands. To remove the help menu repeat the command.

c. Recording

To start the video recorder issue the "RECORD" command or press the "R" key. Repeating this command will pause the recorder. This command is only available if the -r option has been specified.

d. Audio Feedback and Event Notification

Audio Feedback can be activated by the "SOUND" command or pressing "S" on the keyboard. To turn off the audio, repeat the command.

e. Selecting a Vehicle

An individual vehicle is selected by aligning the vehicle in the center of the display (indicated by the +). When the vehicle is centered it will change to a solid green color. Simply looking away from the vehicle will deselect it.

f. Displaying Information

To display currently stored information on any selected vehicle give the verbal command "INFO" or press the "I" key. The information on the vehicle will be shown in the upper right corner of the display and include the vehicle type, country, location, and if an aircraft or missile the altitude. The info display can be deactivated by repeating the command.

g. Attaching and Detaching

You can attach your view to a selected vehicle by issuing the command "ATTACH" or pressing the "A" key. Once attached to a vehicle your movement will be controlled by the movement of the vehicle. Your orientation will remain under your own control so you can still look in any direction. To detach from the vehicle, give the "DETACH" command or press the "D" key. When detached you will be returned to your last position prior to attaching to the vehicle.

h. Enlarging, Removing, and Recovering Vehicles

All vehicles or groups of vehicles can be enlarged by 25 times their normal size by issuing the verbal command "ALL", "AIRCRAFT", "FOES", or "FRIENDS" or pressing the "Q", "P", "X", or "F" keys. Vehicles are restored to their normal size by repeating the command. To remove a selected vehicle or a group of vehicles, issue the "REMOVE UNIT", "REMOVE AIRCRAFT", "REMOVE FOES", or "REMOVE FRIENDS" verbal command or press the "DEL", "B", "V", or "N" key. To recover the removed vehicles give the "RECOVER ALL", "RECOVER AIRCRAFT", "RECOVER FOES", or "RECOVER FRIENDS" command or press the "RET", "G", "J", or "K" key.

i. Special Display Options

Several display options are available to enhance the functionality of SBS. To display active radar envelopes give the "EMITTERS" command or press the "E" key. To display the threat envelopes of SAM and AAA systems issue the command "THREATS" or press the "T" key. Aircraft trails or flight paths covering the previous 10 seconds can be displayed by the "TRAILS" command or the "C" key. Missile tracks for the duration of tracking can be displayed by the verbal command "TRACKS" or the "M" key. To display the launch or first intercept location of all missiles give the verbal command "LAUNCHERS" or press the "L" key. A missile icon will be displayed at the launch

position. All of these display options can be deactivated by repeating the activating command.

j. Defining and Displaying Views

Up to 9 user specified views can be saved and viewed upon request. To define a view give the command "DEFINE VIEW" followed by a number from 1 to 9. The choice of which number is yours. SBS will save all active system status flags and special display options along with your location. The equivalent keyboard commands are the functions keys F1 through F9. The same view can be defined as many times as you like. To display a defined view issue the "DISPLAY VIEW" command followed by the number of the view or press the corresponding number from 1 to 9 on the keyboard. The "0" key is considered the system default view.

k. Exiting SBS

To exit from SBS give the verbal command "EXIT" or press the "Esc" key. Please note there is no fail-safe or second chance on the exit command.

7.0 Limitations

The biggest limitation with SBS is that you must have super user access to run the program. The network manager reads raw Ethernet packets and processes them. IRIX limits this ability to super users only. SBS uses run to completion frame updates and as a result, the more complex the rendered scene, the lower the frame rate. The system is currently limited by the network manager to 500 active vehicles.

8.0 System Commands

VOICE COMMAND	FUNCTION
AIRCRAFT	All Aircraft Vehicles Enlarged Toggle
ALL	All Vehicles Enlarged Toggle
ATTACH	Attach to Selected Vehicle
DETACH	Detach from Vehicle
EMITTERS	Emitter Envelope Display Toggle
EXIT	Exit Program
FOES	Non-friendly Vehicles Enlarged Toggle
FRIENDS	Friendly Vehicles Enlarged Toggle
HELP	Help Menu Toggle
INFO	Vehicle Info Toggle
LAUNCHERS	Missile Launch sites Display Toggle
RECORD	Record Toggle
SOUND	Sound Feedback Toggle
THREATS	Threat Envelope Display Toggle
TRACKS	Missile Tracks Display Toggle
TRAILS	Aircraft Trails Display Toggle
DEFINE VIEW ONE	Define View 1
DEFINE VIEW TWO	Define View 2
DEFINE VIEW THREE	Define View 3
DEFINE VIEW FOUR	Define View 4
DEFINE VIEW FIVE	Define View 5
DEFINE VIEW SIX	Define View 6
DEFINE VIEW SEVEN	Define View 7
DEFINE VIEW EIGHT	Define View 8
DEFINE VIEW NINE	Define View 9
DISPLAY VIEW ZERO	Display Default View
DISPLAY VIEW ONE	Display View 1
DISPLAY VIEW TWO	Display View 2
DISPLAY VIEW THREE	Display View 3
DISPLAY VIEW FOUR	Display View 4
DISPLAY VIEW FIVE	Display View 5
DISPLAY VIEW SIX	Display View 6
DISPLAY VIEW SEVEN	Display View 7
DISPLAY VIEW EIGHT	Display View 8
DISPLAY VIEW NINE	Display View 9
RECOVER ALL	Recover All Vehicles
RECOVER AIRCRAFT	Recover All Aircraft
RECOVER FOES	Recover All Non-friendly Vehicles
RECOVER FRIENDS	Recover All Friendly Vehicles
REMOVE AIRCRAFT	Remove All Aircraft

VOICE COMMAND	FUNCTION
REMOVE FRIENDS	Remove All Friendly Vehicles
REMOVE UNIT	Remove Selected Vehicle

KEY	FUNCTION
ESC	Exit Program
F1- F9	Define View 1-9
0-9	Display View 0-9
A	Attach
B	Remove Aircraft
C	Trails Display Toggle
D	Detach
E	Emitter Display Toggle
F	Friends Enlarged Toggle
G	Recover Aircraft
H	Help Menu Toggle
I	Vehicle Info Toggle
J	Recover Foes
K	Recover Friends
L	Launch sites Display Toggle
M	Missile Tracks Display Toggle
N	Remove Friends
P	Aircraft Enlarged Toggle
Q	All Objects Enlarged Toggle
R	Record Toggle
S	Sound Toggle
T	Threa Display Toggle
V	Remove Foes
X	Foes Enlarged Toggle
Z	Zero Tracking Orientation Angles
DELETE	Remove Unit
RETURN	Recover All
RIGHTARROW	Move Forward
UPARROW	Move Forward
LEFTARROW	Move Backward
DOWNARROW	Move Backward
SPACEBAR	Reset to Default Location

MOUSE BUTTONS

RIGHTMOUSE	Move Forward
MIDDLEMOUSE	Reset to Default Location
LEFTMOUSE	Move Backward

BOOM2M BUTTONS

RIGHTBUTTON	Move Forward
LEFTBUTTON	Move Backward
BOTHBUTTONS	Reset to Default Location

Bibliography

Airey, J. M., Rohlf, J. H., and Brooks, F. P. "Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments." *Computer Graphics Proceedings, SIGGRAPH 90.*

Ascension Technology Corp. *The Bird 6D Graphics Input Device*. Ascension Technology Corporation, Burlington, VT, 1990.

Axt, W. E. "Evaluation of a pilot's line-of-sight using ultrasonic measurements and a helmet mounted display." *Proceedings IEEE National Aerospace and Electronics Conf.* pp 921-927, 1987.

Blau, B., Hughes, C. E., Moshell, J. M., and Lisle, C. "Networked Virtual Environments." *Communications of the ACM*, pp 157-160, 1992.

Booch, G. Software Engineering with Ada. Benjamin/Cummings Publishing Co., 1987.

-----. Software Components with Ada. Benjamin/Cummings Publishing Co., 1989.

Bryson, Steve. "Survey of Virtual Environment Technologies and Techniques." *Implementation of Immersive Virtual Environments Course Notes, SIGGRAPH 92.*

Brunderman, J. *Design and Application of an Object Oriented Graphical Database Management System for Synthetic Environments.* MS thesis, School of Engineering (AU), Air Force Institute of Technology, Wright-Patterson AFB OH, December 1991.

Fake Space Labs. *BOOM2/BOOM2C Operations Manual,* April, 1992.

Ferrin, F. J. "Survey of helmet tracking technologies." *SPIE Vol 1456 Large-Screen Projection, Avionics, and Helmet-Mounted Displays.* pp 86-94, 1991.

Funkhouser, T. A., Sequin, C. H., and Teller, S. J. "Management of Large Amounts of Data in Interactive Building Walkthroughs." *1992 Symposium on Interactive 3D Graphics, special issue of Computer Graphics, pp. 11-20, SIGGRAPH, March 1992.*

Gerken, M. J. *An Event Driven State Based Interface for Synthetic Environments.* MS thesis, School of Engineering (AU), Wright-Patterson AFB OH, December 1991.

Gomaa, H. A. "A Software Design Method for Real-Time Systems." *Communications of the ACM*, September 1984.

-----. " A Software Design Mehtod for Distributed Real-Time Applica[illegible] " *The Journal of Systems and Software*. Vol 9, pp 81-94, 1989.

-----. "Software development of real-time systems." *Communications of t[illegible]CM*, Vol 29, no 7, July 1986.

Hitchner, Lewis E. "Virtual Planetary Exploration: A Very Large Virtual Environment." *Implementation of Immersive Virtual Enviro ments Course Notes, SIGGRAPH 92.*

Hobbs, B. *A User Interface To A True 3-D Display Device*. MS thesis, AFIT/GCS/ENG/92D-06, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1992.

Jacoby, R. H. "Using Virtual Menus in a Virtual Environment." *Implementation of Immersive Virtual Environments Course Notes, SIGGRAPH 92.*

Levi, Shem-Tov, and Agrawala, A. "Real-Time System Design." McGraw-Hill Publishing Co, NY, NY, 1990.

Levit, Creon and Bryson, Steve. "Lessons learned while implementing the virtual windtunnel project," *Implementation of Immersive Virtual Environments Course Notes, SIGGRAPH 92.*

Mackey, R. L. *NPSNET: Hierarchical Dcta Structures for Real-time Three-dimensional Visual Simulation*. MS thesis, Naval Postgraduate School, Monterey California, September 1991.

McDowall, I. E., Bolas, M., Pieper, S., Fisher, S. S., and Humphries, J. "Implementation and Integration of a Counterbalanced CRT-Based Stereoscopic Display for Interactive Viewpoint Control in Virtual Environment Application," *Proc. SPIE Conf. on Stereoscopic Displays and Applications*, J. Merrit and S. Fisher, eds. 1990.

McCarty, D. *Out the Window Cockpit Views for Distributed Simulations*. MS thesis, AFIT/GCS/ENG/92D-22, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1992.

Ming, O. Y., Pi[illegible]ue, M., Hughes, J., Brooks, Jr., F. P. "Using a manipulator for force display in molecular docking." *IEEE Robotics and Automation Conference*. 1988.

Pond, D. L. *A Synthetic Environment for Satellite Modeling and Satellite Orbital Motion*. MS thesis, AFIT/GCS/ENG/92D-22, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1992.

Polhemus. 3SPACE ISOTRAK™ User's Manual. 22 May 1987.

Polhemus, B. President Polhemus Systems. Personal interview. Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 20 January 1993.

Recla, W. F. *A Study in Speech Recognition using a Kohonen Neural Network Dynamic Programming and Multi-Feature Fusion.* MS thesis, AFIT/GE/ENG/89D-41, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1989.

Robinett, Warren, and Rolland, Jannick. "A computational model for the stereoscopic optics of a head-mounted display," *Implementation of Immersive Virtual Environments Course Notes, SIGGRAPH 92,* 1992.

Scarborough, Eric L. *Enhancement of Audio Localization Cue Synthesis by Adding Environmental and Visual Cues.* MS thesis, AFIT/GE/ENG/92D-34, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1992.

Shoemake, K. "Animating Rotations with Quaternion Curves." *Computer Graphics, Proceedings of ACM SIGGRAPH 85,* Volume 19(3), 1985.

Sheasby, S. M. *Management of SIMNET and DIS Entities in Synthetic Environments.* MS thesis, AFIT/GCS/ENG/92D-16, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1992.

Simpson, D. J. *An Application of the Object-Oriented Paradigm to a Flight Simulator.* MS thesis, AFIT/GCS/ENG/91D-22, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1991.

Sutherland, Ivan, "The ultimate display." *Information Processing 1965: Proceedings of IFIP Congress 65.* pp 506-508, 1965.

Switzer, J. C. *A Synthetic Environment Flight Simulator the AFIT Virtual Cockpit.* MS thesis, AFIT/GCS/ENG/92D-17, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1992.

Teller, S. J. and Sequin, C. H. "Visibility Preprocessing For Interactive Walkthroughs." *Computer Graphics, Volume 25, SIGGRAPH 91,* 1991.

Tisdale, D. *Methods for Viewing Radar Cross Sectional Data in 3D.* MS thesis, AFIT/GCS/ENG/92D-22, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1992.

VPL Research. *VPL DataGlove Model 2 Operations Manual,* January, 1989.

Ward, M., Azuma, R., Bennett, R., Gottschalk, S., and Fuchs, H. "A Demonstrated Optical Tracker with Scalable Work Area for Head-Mounted Display Systems." *Communication of the ACM*, pp 43-52, 1992.

Wright C., and Soltz, B. B. *Macintosh Sound Generation Facility 2.0.* Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 15 November 1992.

Zyda, M. J., McGhee, R. B., McConkle, C. M., Nelson, A. H., and Ross, R. S. "A real-time, three-dimensional moving platform visualization tool." *Computers & Graphics*, Vol 14, No. 2, pp 321-333, 1990.

Vita

Rex Gerrin Haddix II was born on September 28, 1955, in Lexington, Kentucky. On April 15, 1975, Rex entered the United States Air Force as a electronic intelligence systems operator (205X0). In May, 1984, Rex received his Bachelor of Science in Computer Science from Chapman College. Rex remained enlisted in the Air Force until 1985, with assignments to Keesler AFB, Mississippi; Offutt AFB, Nebraska, Kadena AB, Japan; and Davis-Monthan AFB, Arizona.

In April, 1985, Rex was commissioned a second lieutenant in the Air Force through the Officer Training School at Lackland AFB, Texas. While on tours of duty in Andersen AFB, Guam, and McCellan AFB, California, Rex completed a Master of Science in Systems Management with Golden Gate University. In 1990, he was accepted to the Air Force Institute of Technology, where he completed his Master of Science degree in Computer Systems in 1993.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1993	MS Thesis

4. TITLE AND SUBTITLE
An Immersive Synthetic Environment for Observation and Interaction with a Large Volume of Interest

5. FUNDING NUMBERS

6. AUTHOR(S)
Capt Rex G. Haddix II

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Wright-Patterson AFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCS/ENG/93M-02

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Joint National Intelligence Development Staff (JNIDS)
4600 Silver Hill Road Washington, D.C. 20389

Defense Advanced Research Projects Agency (DARPA)
3701 N. Fairfax Dr. Arlington VA 22203-1714

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This thesis addresses the initial development of a Synthetic Battlebridge System designed to provide the user with a synthetic three-dimensional view of moving and stationary vehicles dispersed over a hundred thousand cubic mile volume. The system contains provisions to allow a user to view the battlefield either in the Polhemus LookingGlass™ fiber optic based high resolution head mounted display, a standard CRT, or through the Fake Space Labs BOOM2M™ high resolution monochrome display. Users can also video tape a session in the Synthetic Battlebridge System. A voice recognition system provides user interaction to the Synthetic Battlebridge System. Using a positional tracker to monitor position and direction of view, the system allows the user to move around and through the battlefield. The system also allows the user to identify and select up to ten different views of the battlefield.

This is the first year of a continuing effort at the Air Force Institute of Technology (AFIT) towards creating a Synthetic Battlebridge System commanders can use to better understand the spatial orientation and spatial distribution of elements on the battlefield. The system will provide increased situational awareness to assist the commander's critical decision making processes.

14. SUBJECT TERMS	15. NUMBER OF PAGES
	90
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to *stay within the lines* to meet optical scanning requirements.

Block 1. Agency Use Only (Leave blank)

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with ...; Trans. of ...; To be published in .... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities.
NASA	- See Handbook NHB 2200.2.
NTIS	- Leave blank.

Block 12b. Distribution Code.

DOD	- Leave blank.
DOE	- Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	- Leave blank.
NTIS	- Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

*U.S.GPO:1990-273-271